

# Source Localization of Short-Duration Signals Using Wave-Front Curvature

Technical Report

February 28, 1992

Submitted to: Naval Underwater Systems Center

Contract Number: N66604-91-C-7043

Report Authors: Dr. Ben Rosen  
Dr. Michael S. Wengrovitz

Atlantic Aerospace Electronics Corporation

6404 Ivy Lane, Suite 300  
Greenbelt, MD 20770-1406  
301-982-5200

470 Totten Pond Rd.  
Waltham, MA 02154-1905  
617-890-4200

The work presented in this report was performed under Task 2 of the Non-Traditional Signal Project as part of the Submarine/Surface Ship ASW Surveillance Program sponsored by the Antisubmarine Warfare/Undersea Technology Directorate of the Office of Naval Technology: Program Element 0602314N, ONT Block Program NU3B, Project No. RJ14C33, NUWC Job Order A60070, Principal Investigator M.W. Gouzie (Code 2121), NUWC Program Director G.C. Connolly (Code 2193). The sponsoring Activity's Technology Area Manager for Undersea Target Surveillance is T.G. Goldsberry (OCNR 231).

# Contents

|     |  |    |
|-----|--|----|
| 1   | Introduction                                 | 5  |
| 2   | Wave-Front Curvature Range/Bearing Algorithm | 9  |
| 3   | Sensitivity to Time-Delay Measurement Error  | 11 |
| 4   | Sensitivity to Sensor Position Uncertainty   | 21 |
| 5   | Time-Delay Estimation                        | 29 |
| 5.1 | Time-Delay Estimation Algorithms . . . . .   | 29 |
| 5.2 | Correlation Method . . . . .                 | 29 |
| 5.3 | CEP Algorithm . . . . .                      | 32 |
| 5.4 | Full-Wavefield Propagation Models . . . . .  | 34 |
| 6   | Summary                                      | 36 |
| A   | Wave-Front Curvature Equations               | 38 |
| B   | CEP Algorithm                                | 43 |
| C   | Computer Codes                               | 52 |

## List of Figures

|    |  |    |
|----|--|----|
| 1  | Structure of Wave-Front Curvature Range/Bearing Estimator . . . . .          | 8  |
| 2  | Source-Sensor Geometry for a Distorted Linear Array . . . . .                | 9  |
| 3  | Time-Delay Uncertainty Sensitivity (Bearing = 50, Sensor Arm = 60 m) . . .   | 14 |
| 4  | Time-Delay Uncertainty Sensitivity (Bearing = 90, Sensor Arm = 60 m) . . .   | 15 |
| 5  | Time-Delay Uncertainty Sensitivity (Bearing = 50, Sensor Arm = 120 m) . . .  | 16 |
| 6  | Time-Delay Uncertainty Sensitivity (Bearing = 90, Sensor Arm = 120 m) . . .  | 17 |
| 7  | Time-Delay Uncertainty Sensitivity (Bearing = 50, Sensor Arm = 240 m) . . .  | 18 |
| 8  | Time-Delay Uncertainty Sensitivity (Bearing = 90, Sensor Arm = 240 m) . . .  | 19 |
| 9  | Time-Delay Uncertainty Sensitivity (Bearing = 90, Sensor Arm = 240 m) . . .  | 20 |
| 10 | Sensor-Position Uncertainty Sensitivity (Bearing = 50, Sensor Arm = 60 m) .  | 23 |
| 11 | Sensor-Position Uncertainty Sensitivity (Bearing = 90, Sensor Arm = 60 m) .  | 24 |
| 12 | Sensor-Position Uncertainty Sensitivity (Bearing = 50, Sensor Arm = 120 m) . | 25 |
| 13 | Sensor-Position Uncertainty Sensitivity (Bearing = 90, Sensor Arm = 120 m) . | 26 |
| 14 | Sensor-Position Uncertainty Sensitivity (Bearing = 50, Sensor Arm = 240 m) . | 27 |
| 15 | Sensor-Position Uncertainty Sensitivity (Bearing = 90, Sensor Arm = 240 m) . | 28 |
| 16 | Field Propagated by OASES at Multiple Sensors . . . . .                      | 35 |
| 17 | Roots of Determinant: $.001 < \lambda < 8.0$ . . . . .                       | 49 |
| 18 | Roots of Determinant: $10^{-8} < \lambda < 8 \cdot 10^{-3}$ . . . . .        | 50 |
| 19 | Roots of Determinant: $10^{-9} < \lambda < 8 \cdot 10^{-6}$ . . . . .        | 51 |

## List of Tables

|   |   |    |
|---|---|----|
| 1 | Time of Arrival Accuracy Requirements (millisec) for 200 Meter Range Standard Deviation . . . . . | 13 |
| 2 | Sensor Position Accuracy Requirements (m) for 200 Meter Range Standard Deviation . . . . .        | 22 |
| 3 | Bandwidth and Correlation Time . . . . .  | 31 |
| 4 | Mean Delay (and Standard Deviation) (in samples at 100 kHz) . . . . .                             | 32 |

# 1 Introduction

This report describes the work conducted by Atlantic Aerospace Electronics Corporation under contract N66604-91-C-7043 in support of NUSC's efforts to establish passive methodologies for localization of non-cooperating targets emitting high-SNR, short duration signals. Our task was to evaluate the performance of various techniques for locating an acoustic emitter under these conditions using measured times of arrivals at an array of acoustic sensors when little a priori knowledge of the signal statistics is available. This kind of passive localization of underwater targets, termed wave-front curvature ranging, can be accomplished by spatial processing of signals received at multiple hydrophones. However, the successful application of this concept to signals of short duration depends upon accurately estimating the time-difference of arrivals (TDOA) of the signal at several pairs of sensors. Commonly, time-delay measurement is accomplished by cross-correlation of the received signals which is an optimum process only when the signal duration is long compared with the signal correlation time and the true time delay. Short-duration signals make this approach problematical. Another requirement is that accurate knowledge of the location of the sensors be obtained, which holds true for short- as well as long-duration signals. We shall address these issues in this report.

There were three major tasks we set out to perform. The first task was to investigate candidate algorithms, and if necessary, to develop an algorithm to relate the basic measured quantities, the pairwise time-delays between sensors, to the range and bearing of the acoustic emitter. For a linear array, there are analytic expressions for the target range and bearing as functions of the time-difference of arrivals of a signal at multiple sensors located on the array. There are several instances of such algorithms for linear arrays appearing in the literature and we used these as our starting point [4]. While these algorithms are applicable to three-sensor arrays in cases where the array is slightly distorted from a straight line or where the sensors are not equidistant, they are not applicable to both. We extended the applicability of these algorithms to the more general case. It should be noted that the three-sensor case is the minimum number required to estimate range; more sensors could be used. Because the range and bearing equations form a non-linear pair of equations, an iterative algorithm was developed. The algorithm was developed in the SUN MATLAB environment which is conducive to developing and testing algorithms as well as performing numerical experiments of modest size. The algorithm and its application to wave-front curvature ranging is discussed in Section 2 but its full derivation is given in Appendix A.

Our second major task was to study the sensitivity of localization accuracy to time-delay estimation error and sensor location error. It is intuitively clear that inter-sensor distances

(arm-lengths) are an important factor in establishing localization accuracy. Closely-spaced sensors introduce more degradation into the range and bearing estimates and, in the limit of zero arm-length, no range estimate can be derived. Our goal here is to establish, for a few significant arm-lengths and source bearings, the dependence of range variance and range bias upon time-delay accuracy and sensor position uncertainty. These results are range-dependent and therefore, we present the results as a function of range. We try to present the results in such a manner as to clearly show much time delay error is tolerable for a given range error which is really the bottom line. The analyses were accomplished in a Monte-Carlo fashion by calculating the errors induced in range and bearing due to random errors in time-delay and sensor position estimation. These simulation codes were also developed under MATLAB and incorporated the abovementioned range-bearing algorithm. This analysis is discussed in Sections 3 and 4.

Atlantic's third task was to investigate competing time-delay estimation algorithms for short-duration stochastic signals. We considered two such algorithms, the conventional cross-correlator (CCC) and a new maximum likelihood estimation technique (MLM) – an optimum one for short-duration signals. The conventional cross-correlation time-delay estimator is an easy and efficient one to implement in the frequency domain using fast Fourier Transforms. While it may not be an optimum operation in our scenario, its structure is independent of signal statistics and its performance may be more robust than other methods and it may be sufficient in a higher-SNR domain. Numerical experiments were done using this algorithm: a single realization of a stochastic process was added to an ensemble of noise signals and cross-correlated for two sensors. Sample means and standard deviations of the time-delays were then computed to give the desired TDOA statistics. Rather than using MATLAB, this particular experiment was conducted using tools and commands implemented in Atlantic Aerospace's DAT-file language since they required extensive number crunching. These results are discussed in Section 5.

The MLM method does depend upon the signal statistics since it requires computation of the eigenvalues and eigenvectors of the signal covariance function and a subsequent expansion of the data in terms of them. Extensive code was developed at AAEC to determine these quantities but Monte Carlo simulations have yet been done. We hope to perform this in follow-on work. This analysis is discussed in Section 5 and Appendix B.

The analyses presented here are valid only for the simplest of conditions and environments. Only horizontal (i.e. homogeneous) channels are considered with no multipath or reverberation effects. In addition, performance when interfering signals are present or when the signal and noise statistics are non-Gaussian is not addressed.

Figure 1 shows a conceptual view of how a range-bearing estimator might be structured. It consists of four general components: (1) Waveform Segmentation Algorithm, (2) Time-Delay Estimator, (3) Range-Bearing Algorithm and (4) Array-Shape Estimator. The Waveform Segmentation Algorithm detects and isolates candidate energy bearing waveforms from the acoustic sensors output time series. Each set of candidate time-series is passed into the Time-Delay Estimator which determines, by correlation or other technique, the pair-wise time-delays (TDOA)  $\tau_1$  and  $\tau_2$  which are then input to the Range-Bearing Algorithm. Another important input to the Range-Bearing Algorithm are estimates of the locations of the acoustic sensors,  $\hat{\delta}_1$ ,  $\hat{\delta}_2$  and  $\hat{\delta}_3$ . This is usually the output of some array shape algorithm which may use, for example, heading and depth sensors located on the array to estimate its true shape and consequently the location of points on the array. There will necessarily be some residual uncertainty in sensor position. Sensitivity of range and bearing to this uncertainty is discussed below. We will show below how much range uncertainty requirements determine maximum position uncertainty requirements. The work described in this report addresses the functioning of two of the components of this system: the Time-delay Estimator and the Range-Bearing Algorithm. The performance of the other two components needs to be addressed in future work.

## LOCALIZATION ESTIMATOR STRUCTURE

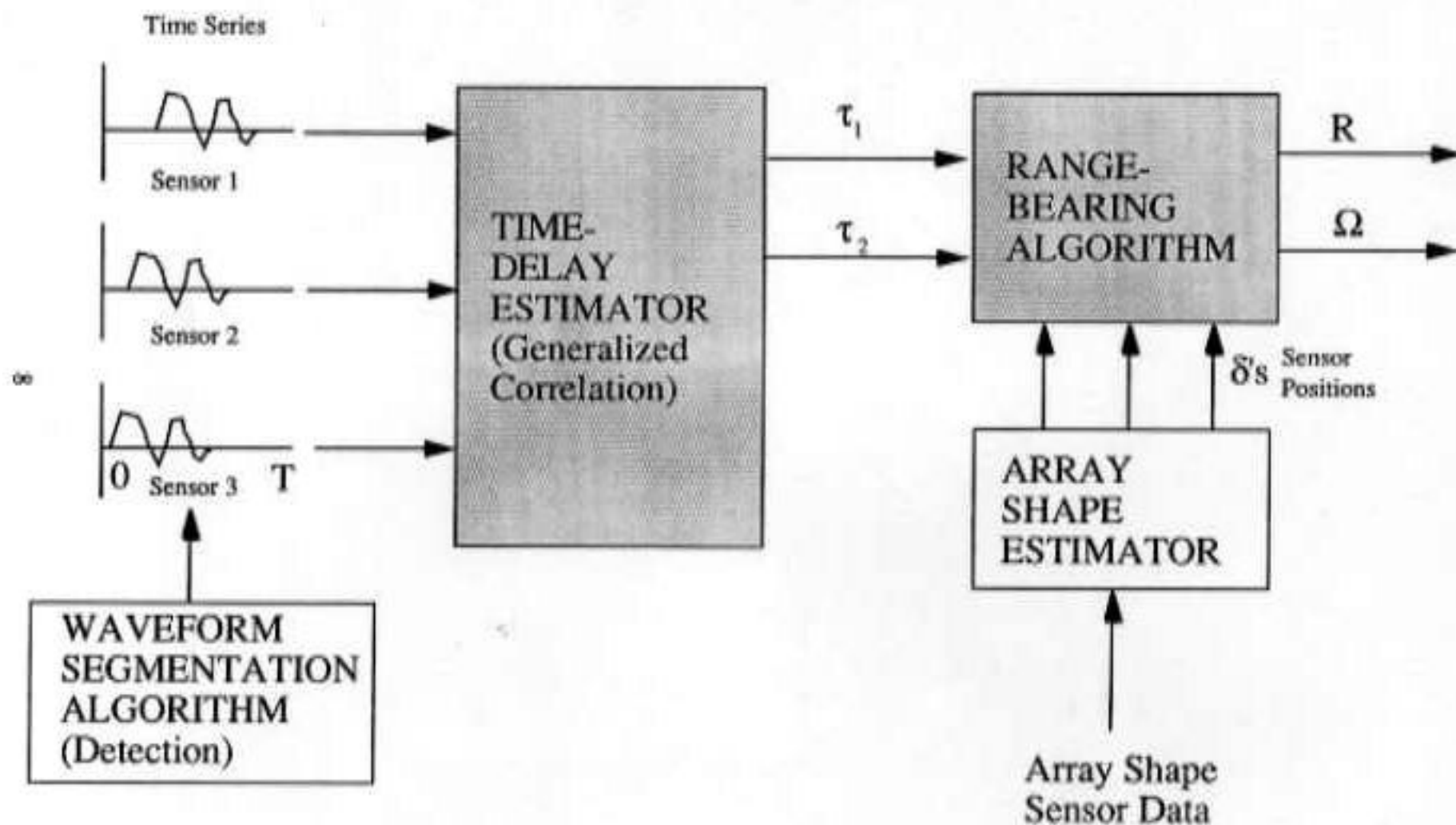


Figure 1. Structure of Wave-Front Curvature Range/Bearing Estimator

## 2 Wave-Front Curvature Range/Bearing Algorithm

The key idea in wave-front curvature localization is to use the differences of time of arrivals (TDOA) of a signal at an array of sensors to compute the range and direction of the signal source. In actuality, it is no more than the concept of triangulation and can in principle be solved for an arbitrary number of sensors (three or more) in any configuration while using two sensors can only estimate bearing. The analyses presented in this report are based upon a three-element near-linear array as shown in Figure 2. In this case, analytic expressions for range and bearing can be derived. If the times-of-arrival of the signals at the three sensors are denoted by  $t_1$ ,  $t_2$  and  $t_3$ , the fundamental measured quantities are the time-difference of arrivals between two pairs of sensors,  $\tau_1 = t_1 - t_2$  and  $\tau_2 = t_2 - t_3$ . The nominal sensor geometry is defined by the vectors  $\vec{L}_1$  and  $\vec{L}_2$  which point from the middle sensor (#2) to the forward (#1) and aft sensors (#3) respectively. The colinearity condition is simply expressed by  $\vec{L}_2 = -a\vec{L}_1$ . No assumptions are made concerning the relative lengths of the array arms;  $a$  is one for equal length arms. Under the best of towing conditions, some distortion of the array will occur. The quantities  $\vec{\delta}_1$ ,  $\vec{\delta}_2$  and  $\vec{\delta}_3$  define the deviations or distortions of the sensors from their nominal colinear positions.

This section simply states here the equations relating TDOA and range and bearing; the full derivation is given in Appendix A. Range ( $R$ ) and bearing ( $\Omega$ ), defined with respect to the nominal position of the middle sensor (#2) and the nominal array direction, are given by

$$R = \frac{N(\tau_1, \tau_2) + \vec{L}_1 \cdot \vec{\beta} + c\hat{R} \cdot \vec{\delta}_2 \Delta\tau}{(2c\Delta\tau + \hat{R} \cdot \vec{\alpha})} \quad (2.1)$$

$$\begin{aligned} \cos \Omega = & -\frac{c}{L_1} \frac{a}{a+1} \left( \tau_1 + \frac{1}{a^2} \tau_2 \right) - \frac{c^2}{2RL_1} \frac{a}{a+1} \left( \tau_1^2 - \frac{1}{a^2} \tau_2^2 \right) + \frac{a}{a+1} \frac{\vec{L}_1}{R} \cdot \vec{\xi} \\ & + \frac{c}{RL_1} \frac{a}{a+1} \left( \tau_1 + \frac{1}{a^2} \tau_2 \right) \hat{R} \cdot \vec{\delta}_2 - \frac{a}{a+1} \frac{\hat{R}}{L_1} \cdot \vec{\gamma} \end{aligned} \quad (2.2)$$

where  $\vec{\alpha}$ ,  $\vec{\beta}$ ,  $\vec{\gamma}$  and  $\vec{\xi}$  are certain linear combinations of the distortion vectors. These and other quantities are defined by:

$$N(\tau_1, \tau_2) = \frac{1}{2} L_1^2 \left\{ 1 - \left( \frac{c\tau_1}{L_1} \right)^2 + a \left( 1 - \left( \frac{c\tau_2}{L_2} \right)^2 \right) \right\} \quad (2.3)$$

$$\vec{\alpha} = \vec{\delta}_1 - \vec{\delta}_2 + \frac{1}{a} (\vec{\delta}_3 - \vec{\delta}_2) \quad (2.4)$$

$$\vec{\beta} = \vec{\delta}_3 - \vec{\delta}_1 \quad (2.5)$$

$$\bar{\tau} = (\bar{\delta}_1 - \bar{\delta}_2) - \frac{1}{a^2} (\bar{\delta}_3 - \bar{\delta}_2) \quad (2.6)$$

$$\bar{\xi} = \left( \bar{\delta}_1 + \frac{1}{a} \bar{\delta}_3 \right) \quad (2.7)$$

An important quantity appearing in the denominator of the range equation is the double time-difference

$$\Delta\tau = \tau_1 - \frac{1}{a} \tau_2 \quad (2.8)$$

Since both terms in  $\Delta\tau$  are very nearly equal, their difference is a very small quantity and gets smaller as (1) the range of the source increases and (2) bearing approaches endfire. This limits the usefulness of this concept under those conditions. It is not the fault of the particular algorithm but rather a fundamental inability to measure these quantities. The second limitation is somewhat easier to overcome by turning the array. The first limitation can be remedied by increasing the inter-sensor distance. The small size of  $\Delta\tau$  makes range determination very sensitive to TDOA measurement errors and errors in the measured sensor position.

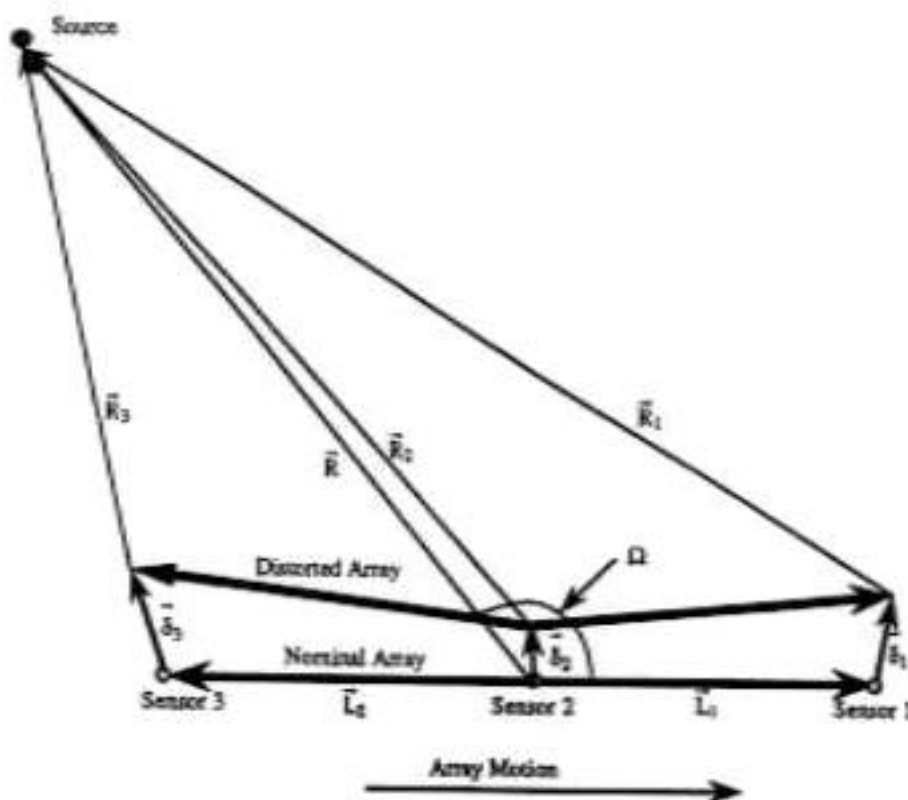


Figure 2: Source-Sensor Geometry for a Distorted Linear Array

### 3 Sensitivity to Time-Delay Measurement Error

This section analyzes the sensitivity of range and bearing estimates to uncertainty in the measurement of the time-difference of arrivals (TDOA) of the emitted signal at the array sensors. Small uncertainty in TDOA measurements can lead to significant error in the estimated range of the emitter which gets worse as the target range increases. This is due to the non-linear inverse relation between  $R$  and  $\Delta\tau$ . This non-linear relation between range and TDOA also makes it more convenient to study sensitivity issues using Monte Carlo simulation rather than to attempt an analytic formulation.

The Monte Carlo technique involves computing the true times-of-arrival (TOA's) of the signal at each acoustic sensor using a prescribed source-sensor geometry. The measured TOA is modeled as the true TOA plus a component which is a sample from a zero-mean Gaussian random process. The random components for each sensor are independent, identically-distributed (i.i.d.) and the mean of the measured TOA is the true TOA. The variance of each sensor's TOA is the independent varying parameter for each experiment and the sample mean and variance of the computed range and bearing are determined as a function of TOA variance. Since range and bearing are actually functions of time-difference of arrival statistics, the TOA variances need to be converted to TDOA variance. This is simple in the Gaussian noise case since a linear combination of Gaussian processes is itself Gaussian and the TDOA variance  $\sigma_\tau^2$  is related to the TOA variance  $\sigma_t^2$  by  $\sigma_\tau^2 = 2\sigma_t^2$ .

Because of the non-linear relation between the TDOA and range and bearing, range and bearing statistics are non-Gaussian, but their sample means and sample variances are still meaningful expressions of uncertainty. Notice that because of the non-linearity, the mean value of the computed ensemble of ranges is biased.

Monte Carlo experiments were performed for several different scenarios: five different target ranges (1 km, 2 km, 5 km, 7.5 km and 10 km) and two different source bearings (50 degrees and 90 degrees (broadside)). This was done for three different array arm lengths (60m, 120m and 240m), totalling 30 different scenarios. For each scenario, a computer experiment was conducted where the TOA standard deviation ( $\sigma_t$ ) varied continuously from 2 $\mu$ sec to 200 $\mu$ sec. For each  $\sigma_t$ , 4000 random TOA's were generated at each of three sensors using MATLAB's RAND function. Time-delays were calculated and passed into the range-bearing algorithm to give a computed range and bearing. From these 4000 samples, mean range, range standard deviation, mean bearing and bearing standard deviation were computed.

From the definition of  $\Delta\tau = \tau_2 - \tau_1$ , it can happen that  $\Delta\tau < 0$  for sufficiently large vari-

ances, a meaningless result. In order to provide meaningful statistics on range, we eliminated calculated ranges corresponding to very small or negative values of  $\Delta r$ . (Very small values of  $\Delta r$  give rise to very large ranges.) The choice of this cutoff is somewhat arbitrary when no a priori information about target location is given but 40 km was chosen as convenient. When some a priori estimate of target location is given, a more precise range gate can be used. Finally, in this simulation, the array was assumed to be colinear with no perturbations of the sensors from their nominal positions.

The basic results are presented in a series of graphs in Figures 3 to 8. Each plot, labelled by the sensor arm length and source bearing, shows how range error (bias and standard deviation) grow by orders of magnitude as TOA uncertainty increases. It is also clear that the range variance dominates range bias. (Plots were drawn on a logarithmic scale because of the large range of values of all the quantities. In order to present the biases on a logarithmic scale, their absolute values were used, hence the cusp in the bias plots where they changed sign). An example of worst-case localization is shown in Figure 3 for an array with a 60m sensor-arm and a source bearing of 50 degrees. In order to achieve a localization error of 200 meters for a target at 10 km, TOA resolution should be within .06 millisecc. Best-case localization is shown in Figure 8 where an array with a 240 meter arm-length is used and the source bearing is 90 degrees.

The final figure in this section (Figure 9) shows a worst-case bearing-error result and indicates that bearing error is better behaved than range error and only grows appreciably for TOA's greater than one millisecond.

Another way to state the results of this section is to establish a localization requirement and to ask what measurement uncertainty is needed to achieve that requirement. As an example, if one adopts 200 meters as a nominal localization goal, Table 1 shows the TDOA measurement accuracy (in milliseconds) required to meet this localization goal. As one can readily see from this table, the requirements for 200 meter localization at 10 km range is quite stringent. In order to localize a source at 10 km using a 240m array, one has to resolve the TDOA to within 30 $\mu$ sec.

| Array Length | Incident Angle | Target Range |      |      |        |       |
|--------------|----------------|--------------|------|------|--------|-------|
|              |                | 1 km         | 2 km | 5 km | 7.5 km | 10 km |
| 60 m         | 50°            | .152         | .040 | .007 | .003   | .002  |
|              | 90°            | .252         | .066 | .012 | .005   | .003  |
| 120 m        | 50°            | .583         | .181 | .028 | .012   | .007  |
|              | 90°            | .964         | .298 | .047 | .020   | .012  |
| 240 m        | 50°            | 2.232        | .689 | .109 | .047   | .028  |
|              | 90°            | 3.693        | 1.14 | .180 | .078   | .047  |

Table 1: Time of Arrival Accuracy Requirements (millisec) for 200 Meter Range Standard Deviation

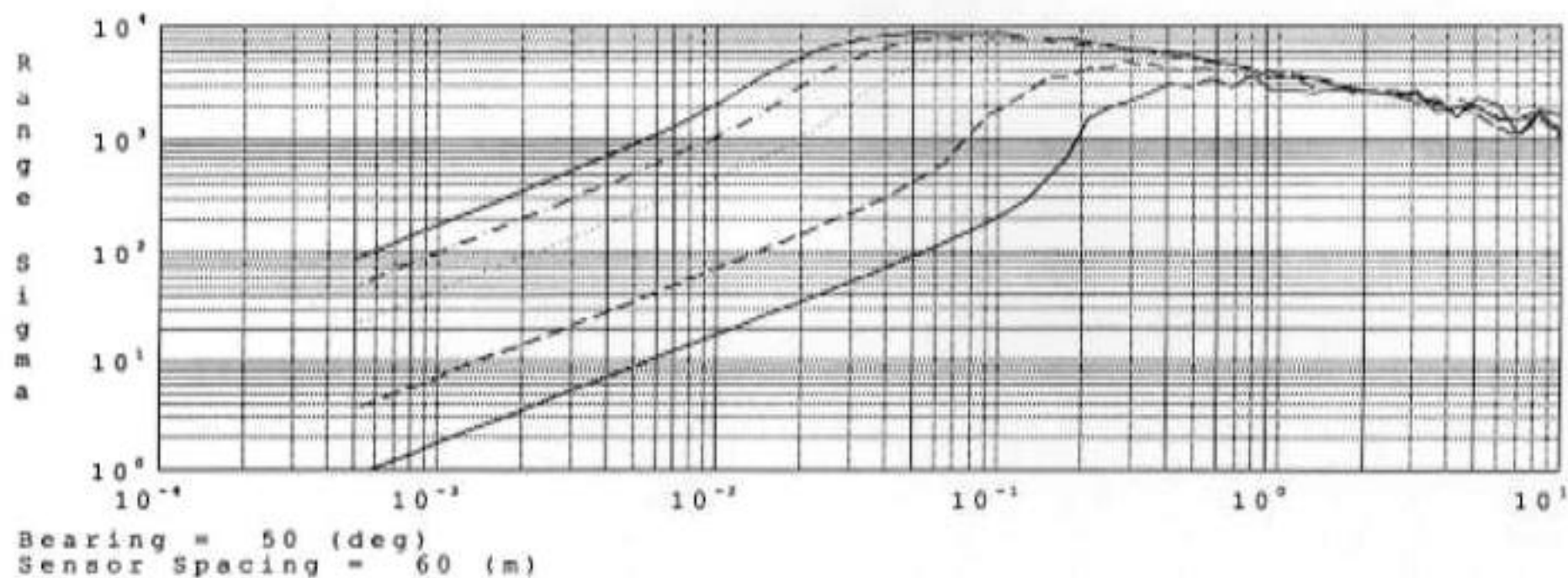
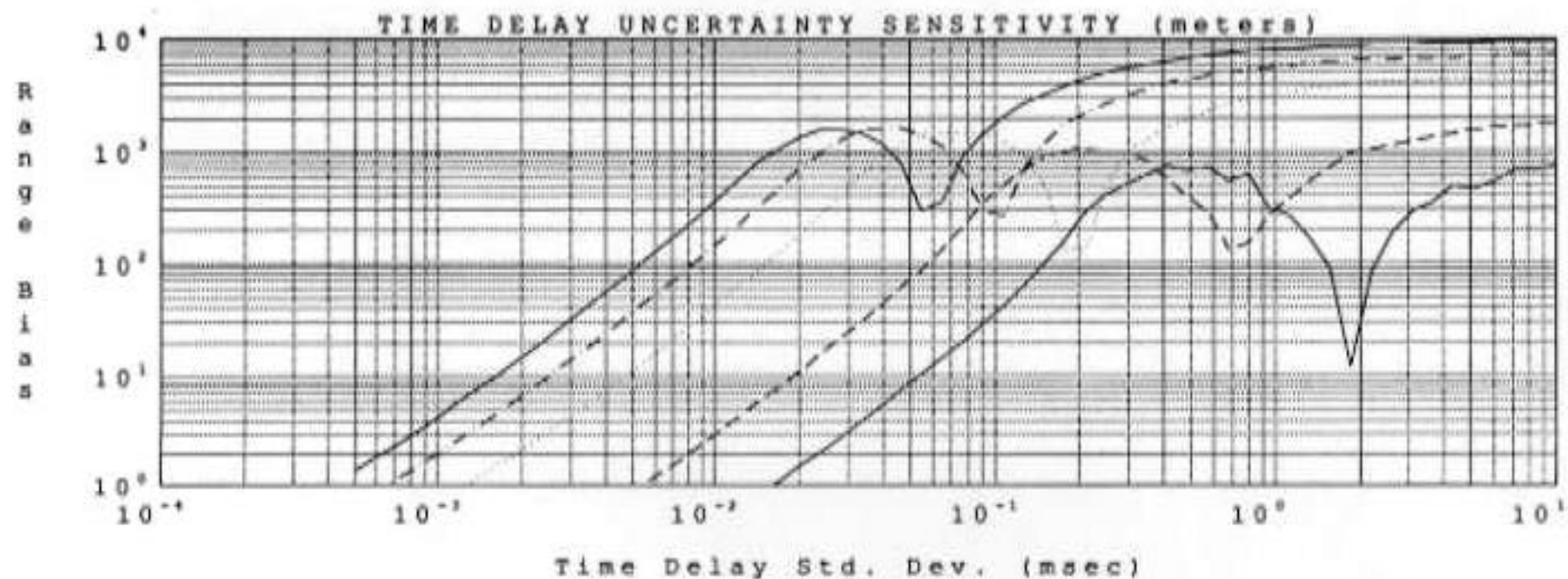
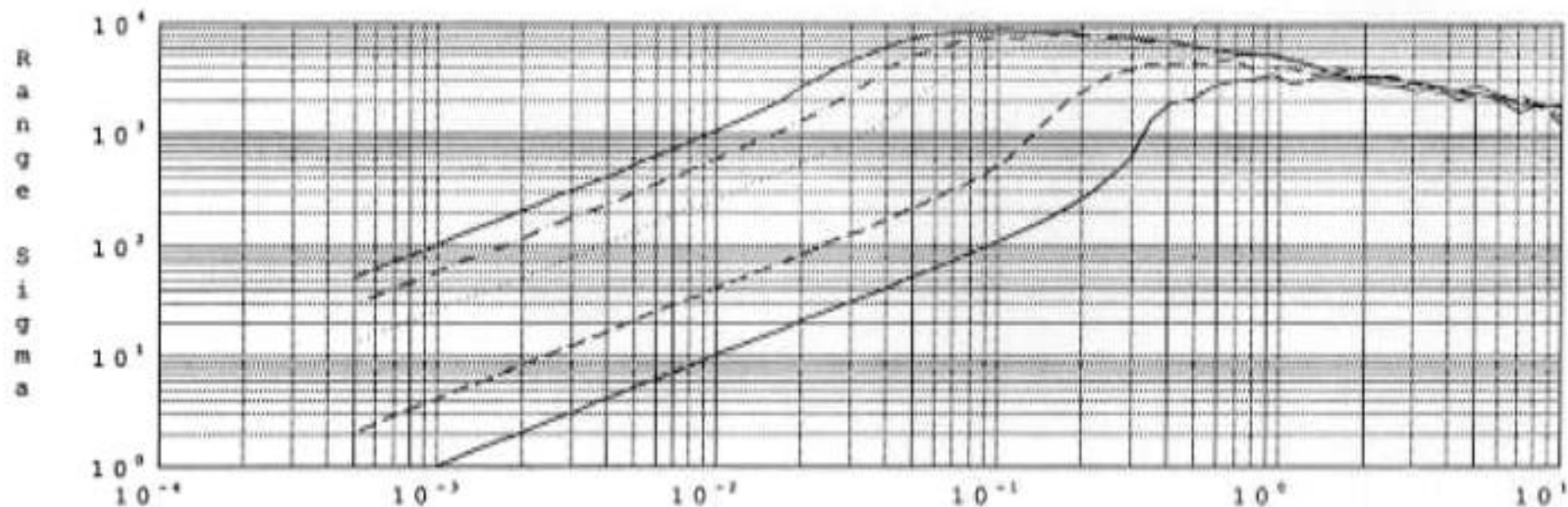
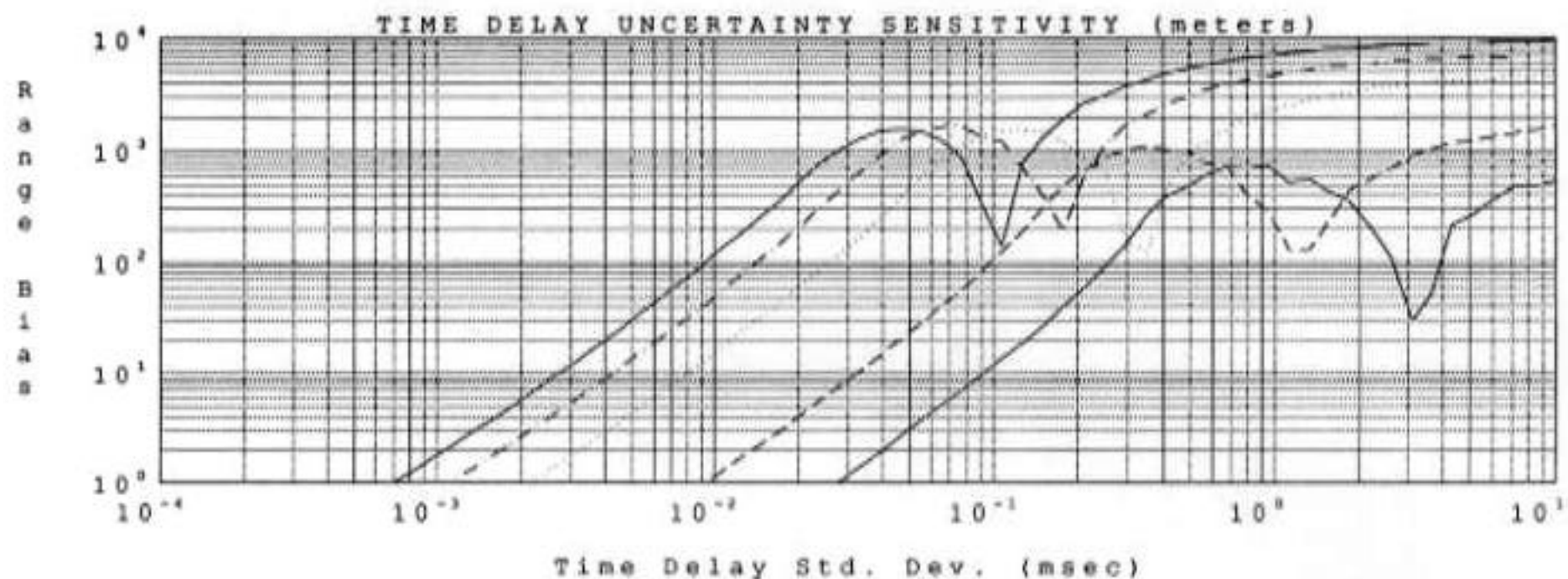


Figure 3: Time-Delay Uncertainty Sensitivity (Bearing = 50, Sensor Arm = 60 m)



Bearing = 90 (deg)  
Sensor Spacing = 60 (m)

Figure 4: Time-Delay Uncertainty Sensitivity (Bearing = 90, Sensor Arm = 60 m)

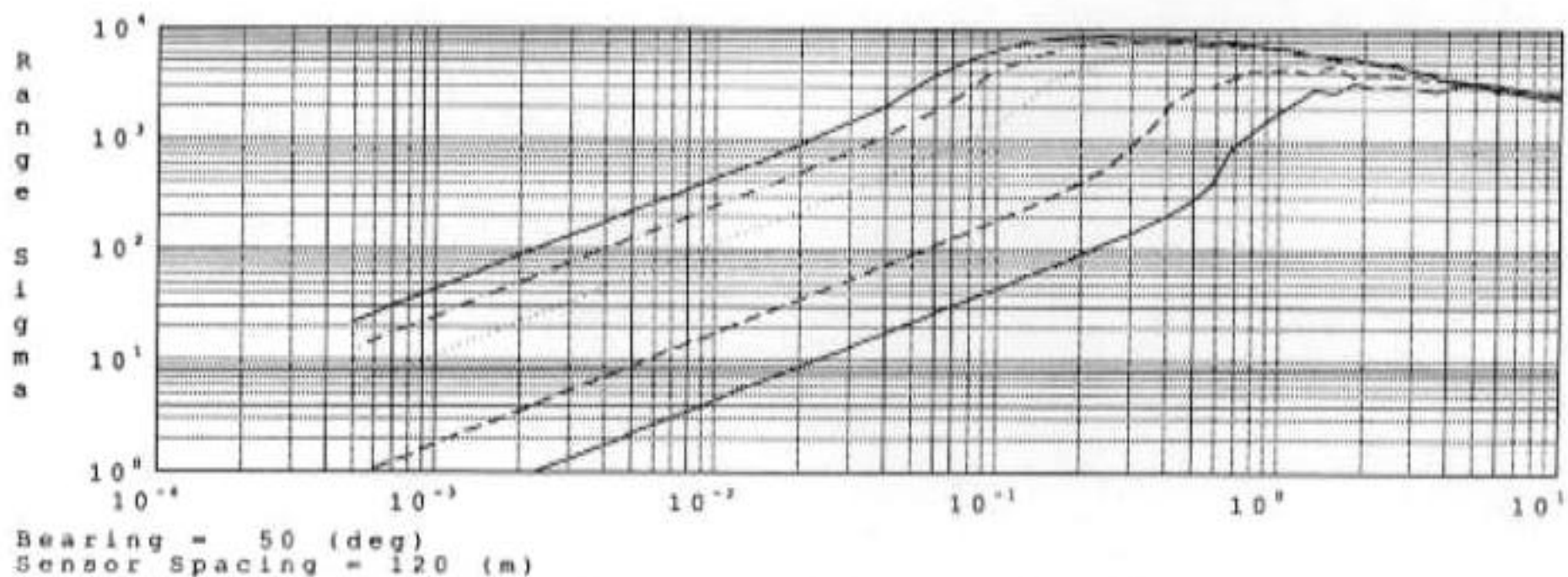
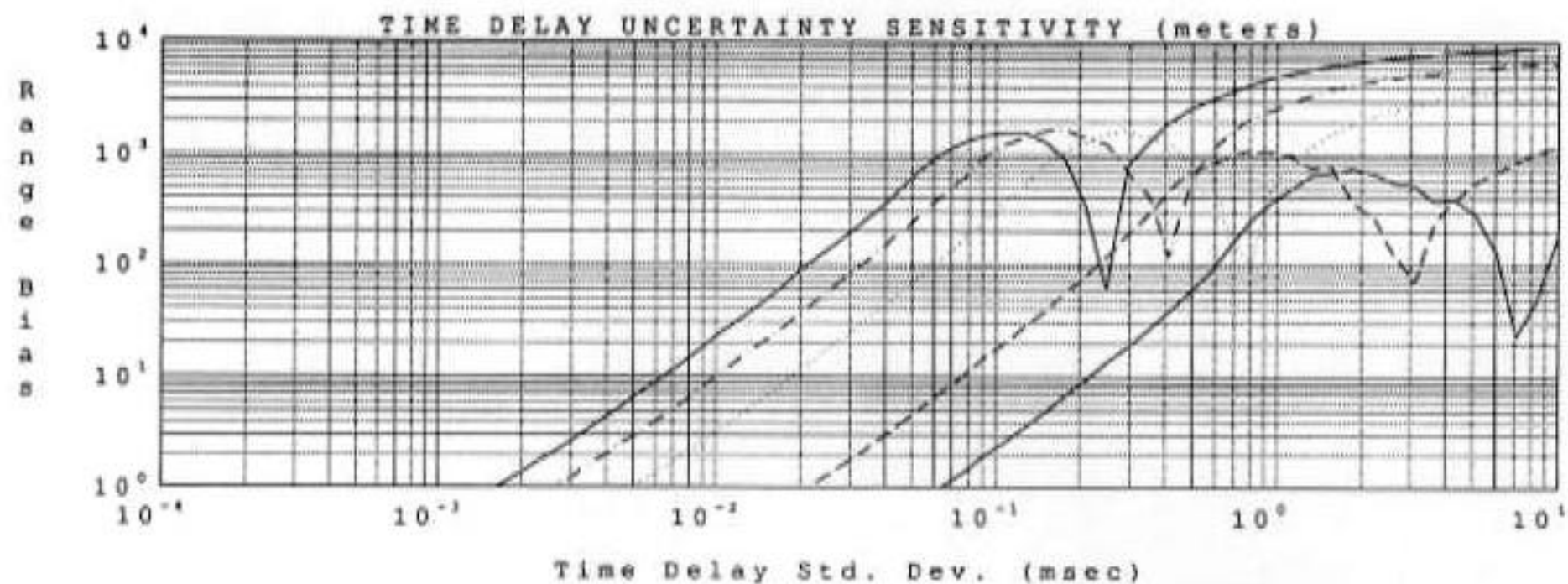
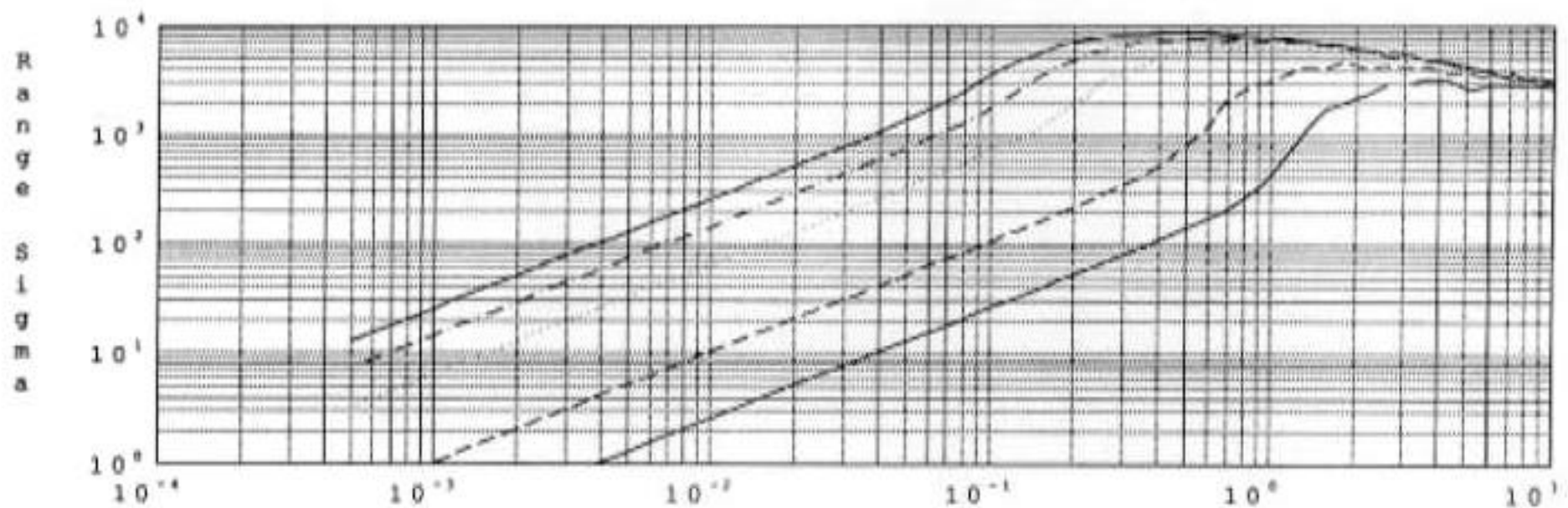
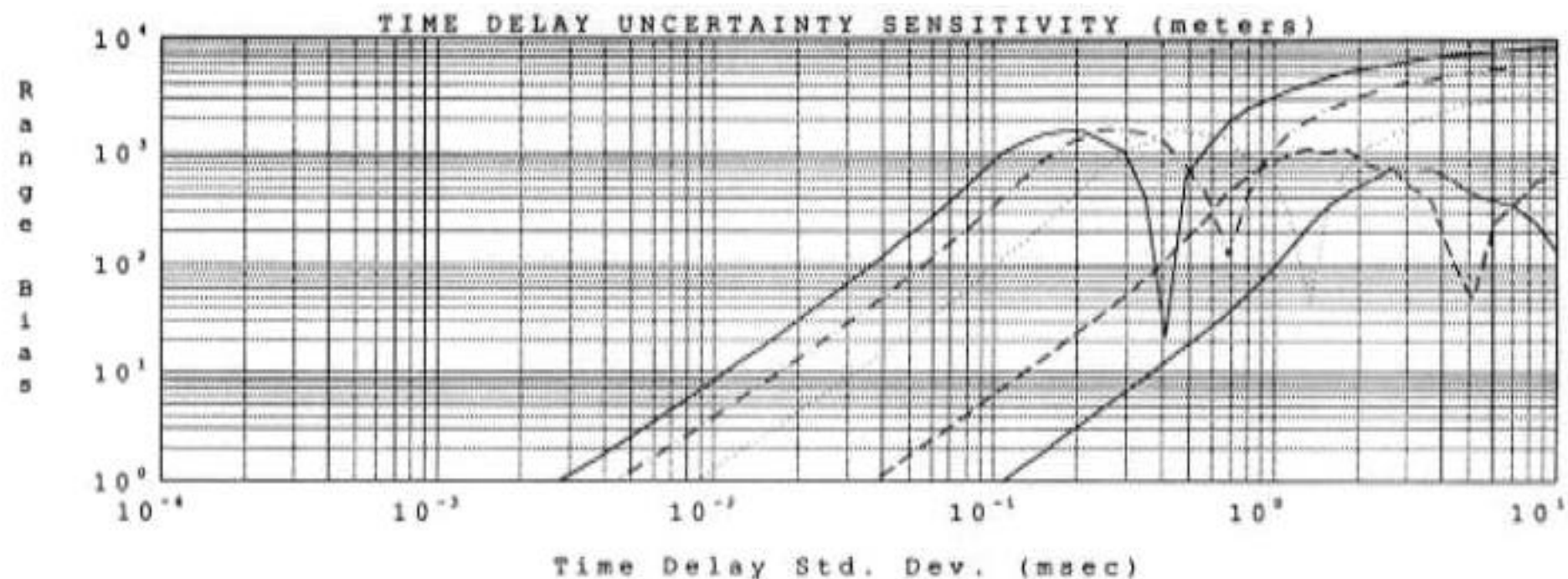


Figure 5: Time-Delay Uncertainty Sensitivity (Bearing = 50, Sensor Arm = 120 m)



Bearing = 90 (deg)  
Sensor Spacing = 120 (m)

Figure 6: Time-Delay Uncertainty Sensitivity (Bearing = 90, Sensor Arm = 120 m)

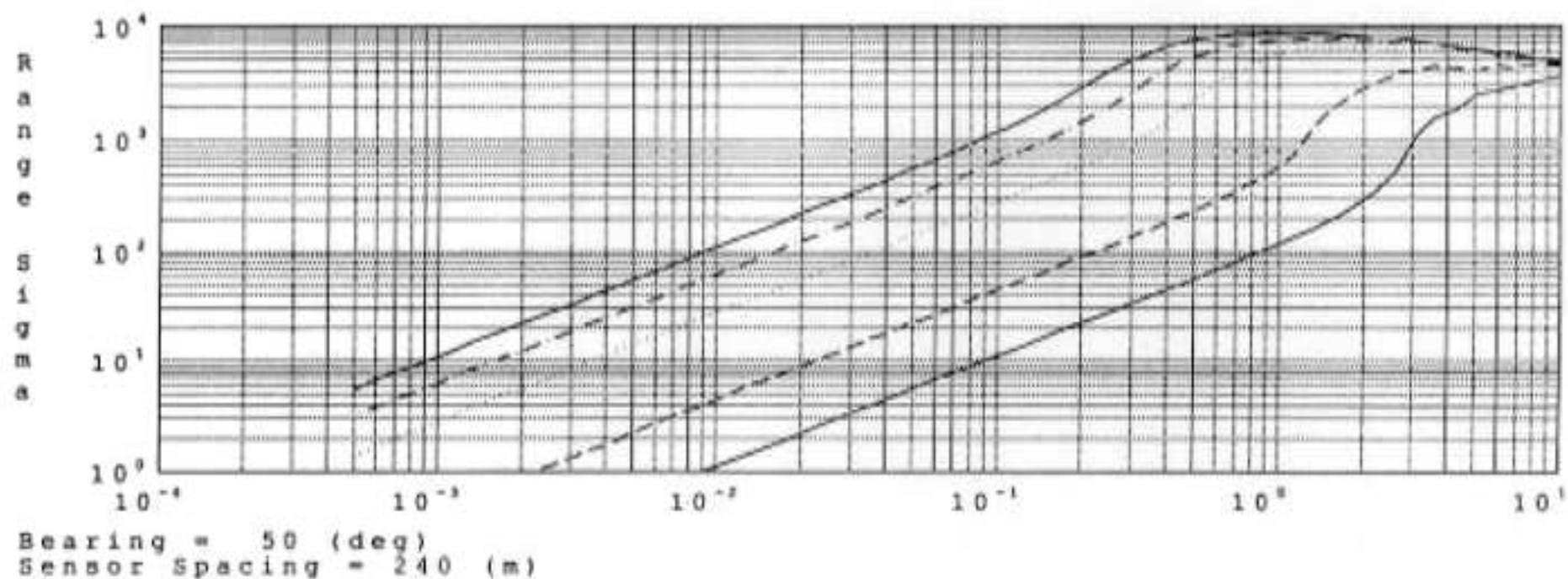
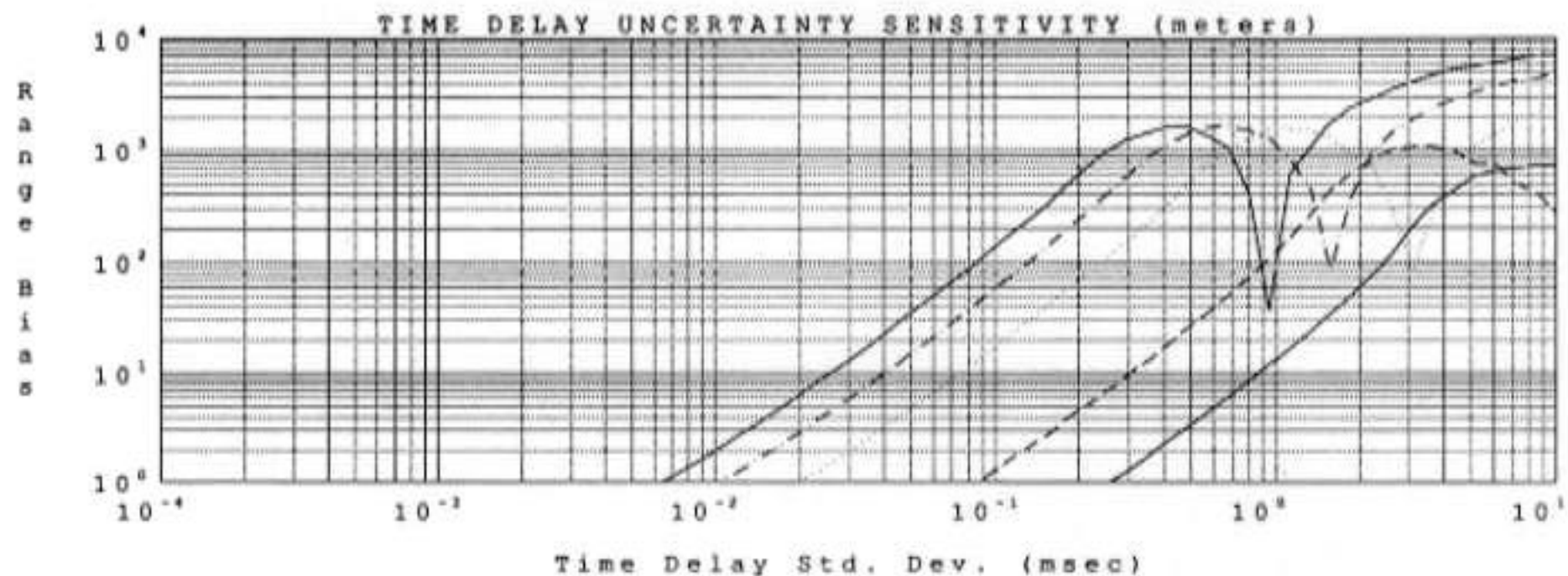
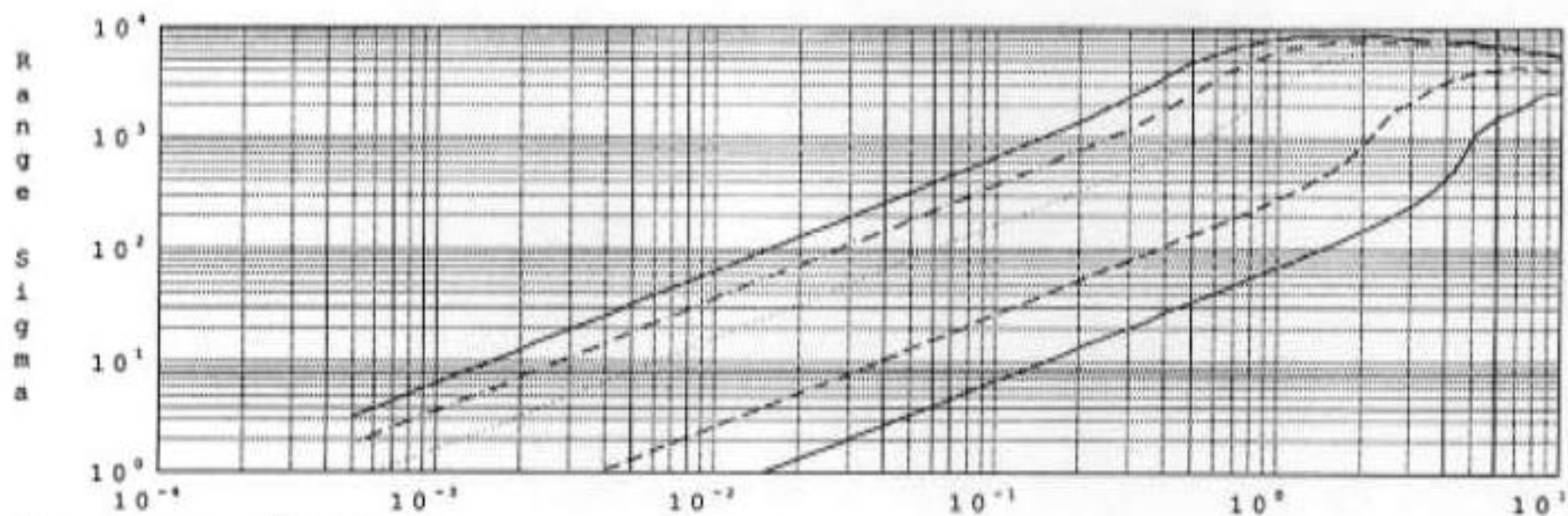
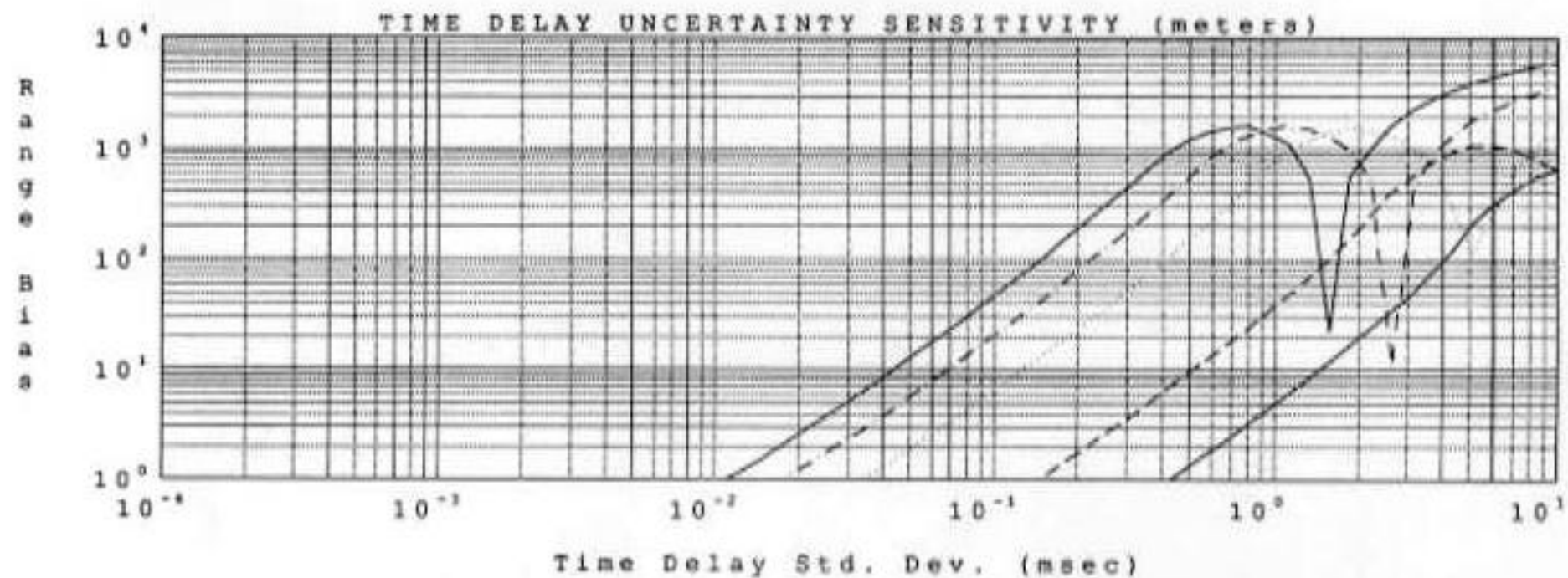


Figure 7: Time-Delay Uncertainty Sensitivity (Bearing = 50, Sensor Arm = 240 m)



Bearing = 90 (deg)  
Sensor Spacing = 240 (m)

Figure 8: Time-Delay Uncertainty Sensitivity (Bearing = 90, Sensor Arm = 240 m)

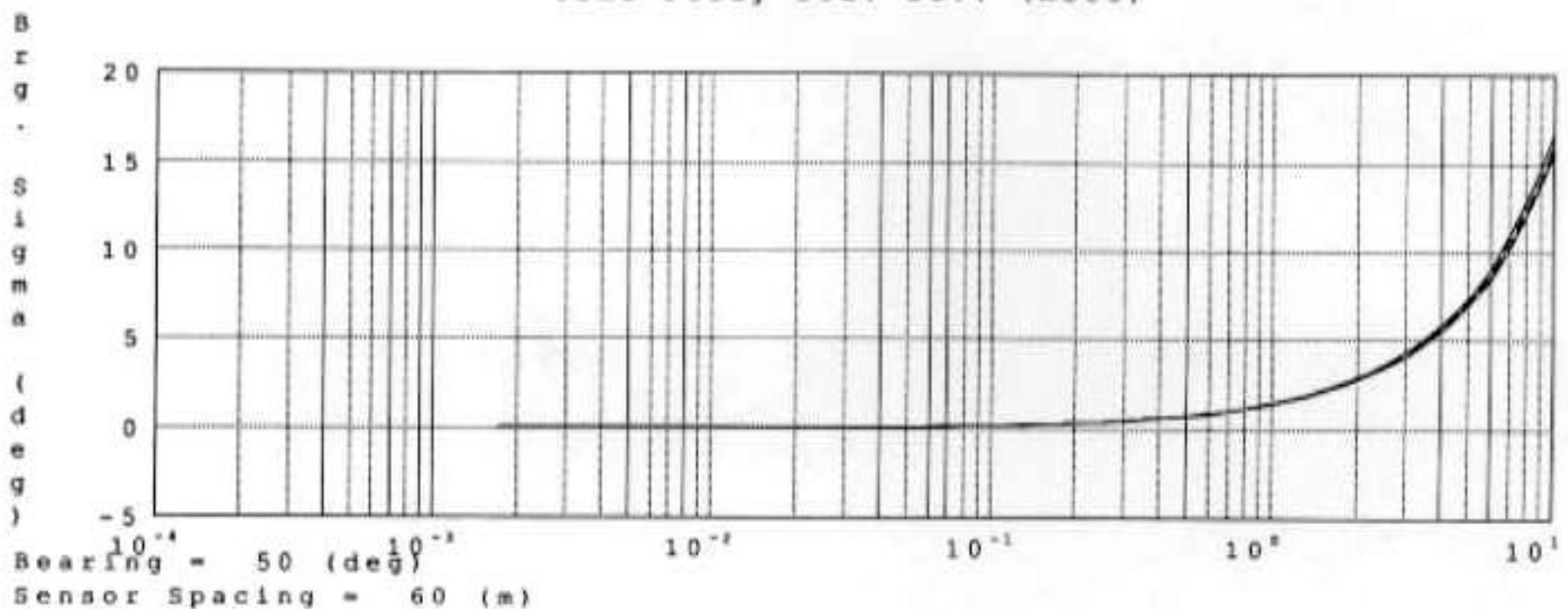
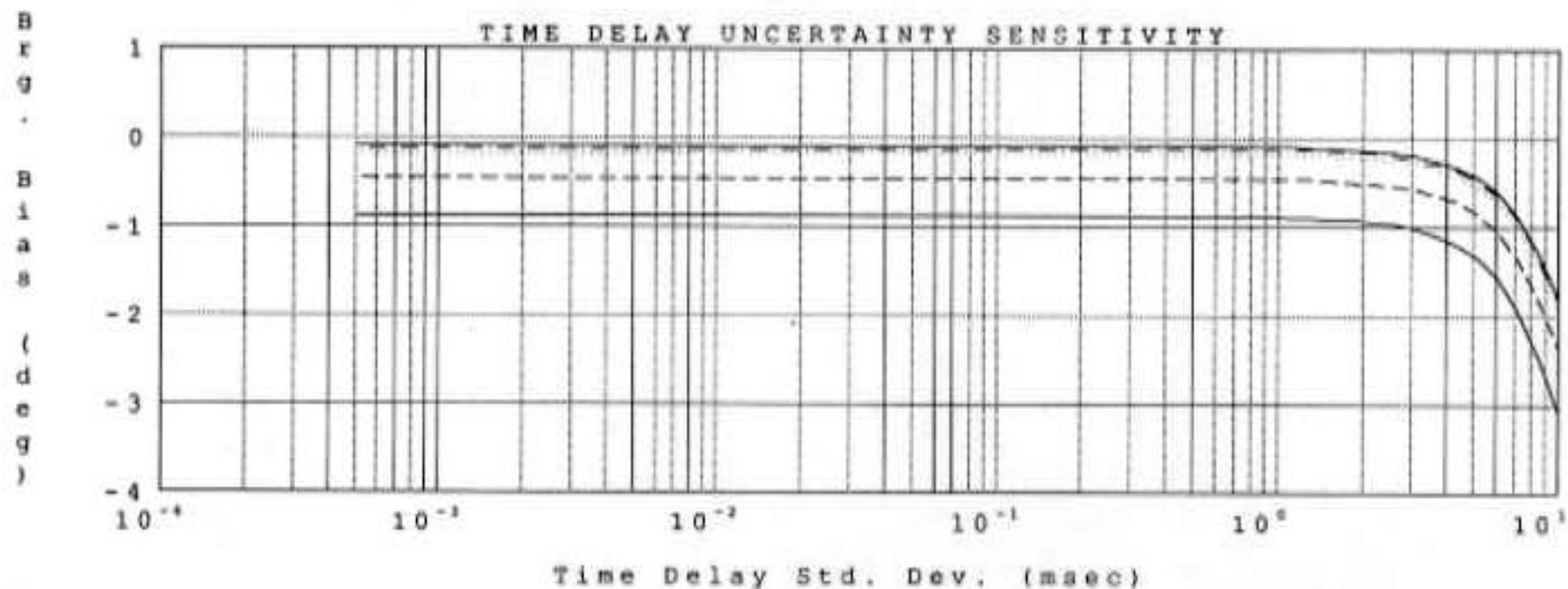


Figure 9: Time-Delay Uncertainty Sensitivity (Bearing = 90, Sensor Arm = 240 m)

## 4 Sensitivity to Sensor Position Uncertainty

Just as errors in the measured times of arrivals of the received signals can significantly affect the estimated range of the emitter, so do errors in the assumed positions of the sensors. In this section, this effect is quantified by modeling the sensor positions as fixed but having an added random vector component which is assumed to be a zero mean Gaussian process. Inspection of Equation 2.1 shows that the most significant contribution to range error comes from the term  $\hat{R} \cdot \vec{\alpha}$  appearing in the denominator where  $\hat{R}$  is a unit vector pointing towards the source and  $\vec{\alpha}$  is a linear combination of sensor deviations defined in Section 2 (It is easy to show that  $\vec{\alpha} = 0$  when the distortion is due entirely to a translation or rotation of the entire array;  $\alpha$  is non-zero when the array is bent). Errors in this term contribute to range localization error in exactly the same way as errors in  $\Delta\tau$  for the time-delay sensitivity issue. In fact, looking at the denominator term  $c\Delta\tau + \hat{R} \cdot \vec{\alpha}$ , one can estimate what order of magnitude errors in  $\Delta\tau$  and  $\vec{\alpha}$  are equivalent. A one millisecond error in time is approximately equivalent to a one meter error in position.

A set of simulation experiments were performed to determine the effect of sensor uncertainty upon range estimation using the same scenarios used in the time-delay experiments. The array was nominally straight and deviations from co-linearity were assumed to be due to random perturbations of each sensor. The perturbations were represented by zero-mean Gaussian processes which were independent from sensor to sensor; the standard deviations of each of these process were the same. Note that  $\vec{\alpha}$  is also a zero-mean Gaussian random process. For each value of sensor standard deviation  $\sigma_s$ , 4000 experiments were done and sample mean range and standard deviation were computed as a function of  $\sigma_s$ .

The following series of plots (Figures 10 through 15) present range error as a function of uncertainty in sensor position. The error consists of a bias term which is inherently due to the non-linear relation between the range and sensor position and the range standard deviation. As we noted above, the results are not unexpectedly similar to the results obtained for the TDOA measurement uncertainty.

Again, given some a priori requirement on range localization, one can ask what measurement uncertainty is needed to achieve that requirement. Adopting 200 meters as a nominal localization goal, Table 2 shows the sensor position measurement accuracy (in meters) required to meet this localization goal. For example, a 1cm accuracy is needed to locate a target at 10km using a 120m array.

| Array Length | Incident Angle | Target Range |       |      |        |       |
|--------------|----------------|--------------|-------|------|--------|-------|
|              |                | 1 km         | 2 km  | 5 km | 7.5 km | 10 km |
| 60 m         | 50°            | .178         | .046  | .007 | .003   | .002  |
|              | 90°            | .249         | .077  | .012 | .006   | .003  |
| 120 m        | 50°            | .682         | .178  | .033 | .014   | .007  |
|              | 90°            | 1.128        | .295  | .055 | .024   | .012  |
| 240 m        | 50°            | 2.611        | .682  | .127 | .055   | .028  |
|              | 90°            | 4.32         | 1.128 | .211 | .091   | .055  |

Table 2: Sensor Position Accuracy Requirements (m) for 200 Meter Range Standard Deviation

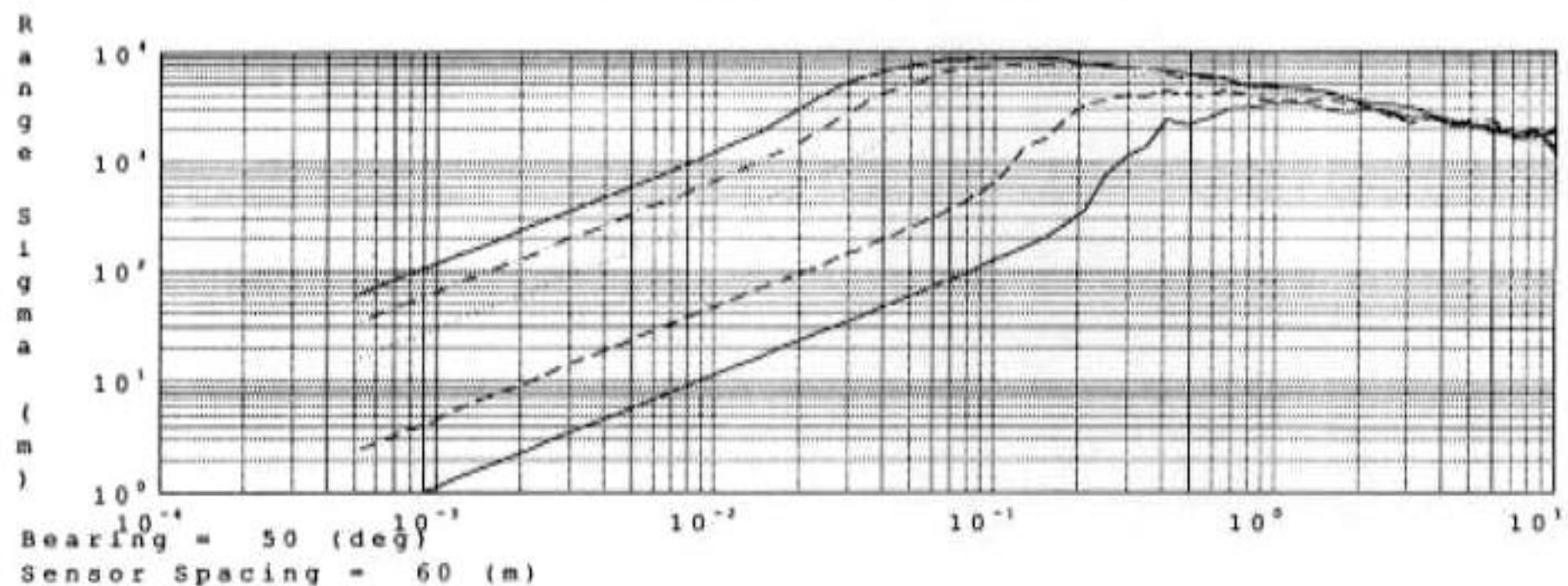
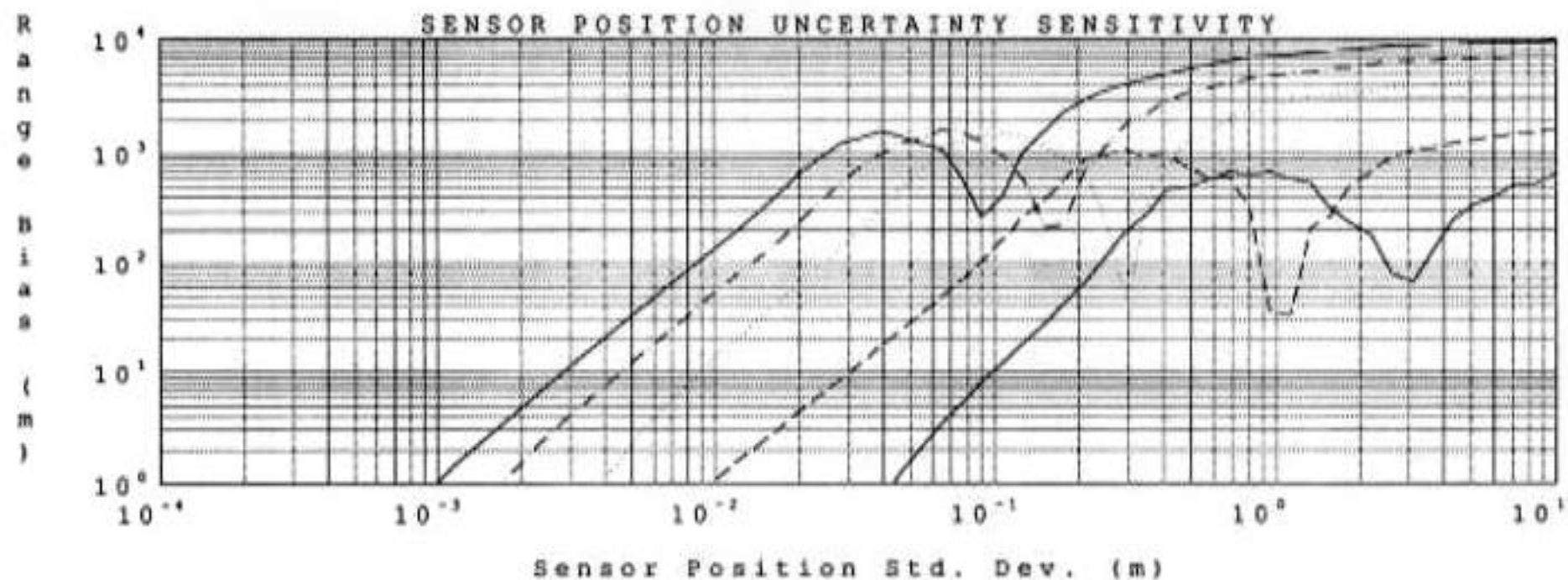


Figure 10: Sensor-Position Uncertainty Sensitivity (Bearing = 50, Sensor Arm = 60 m)

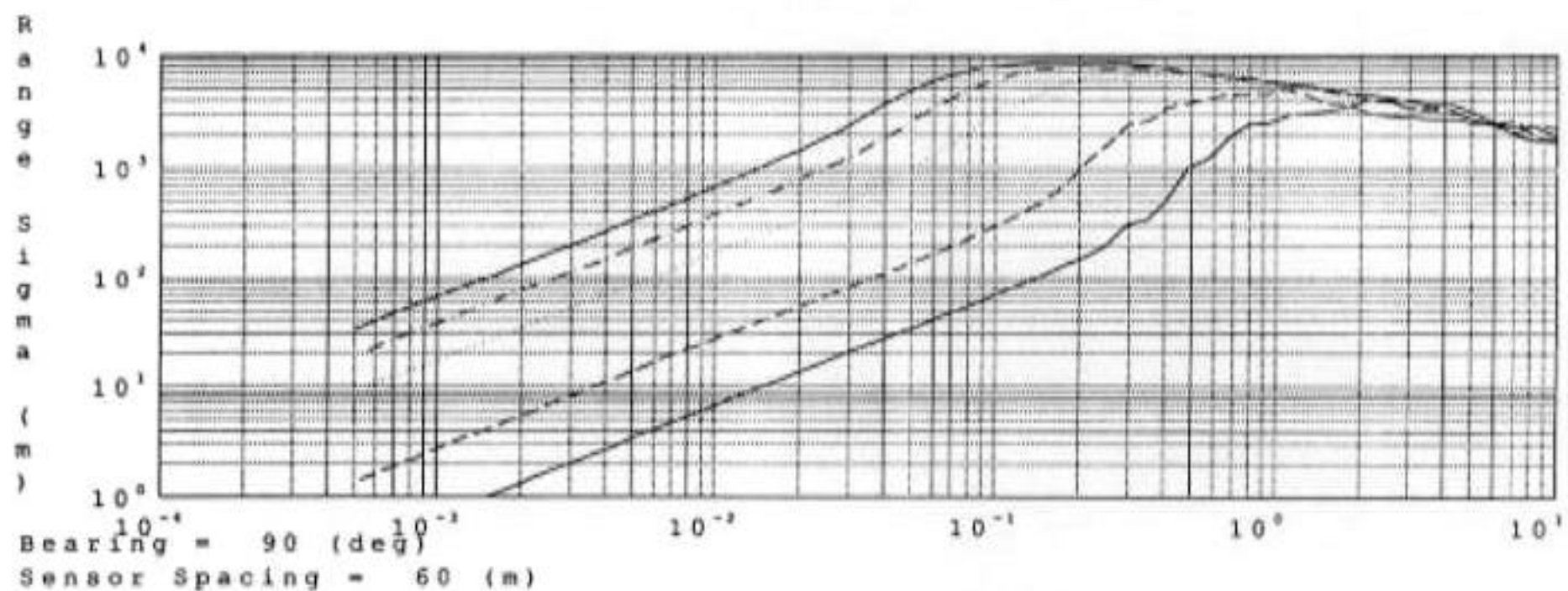
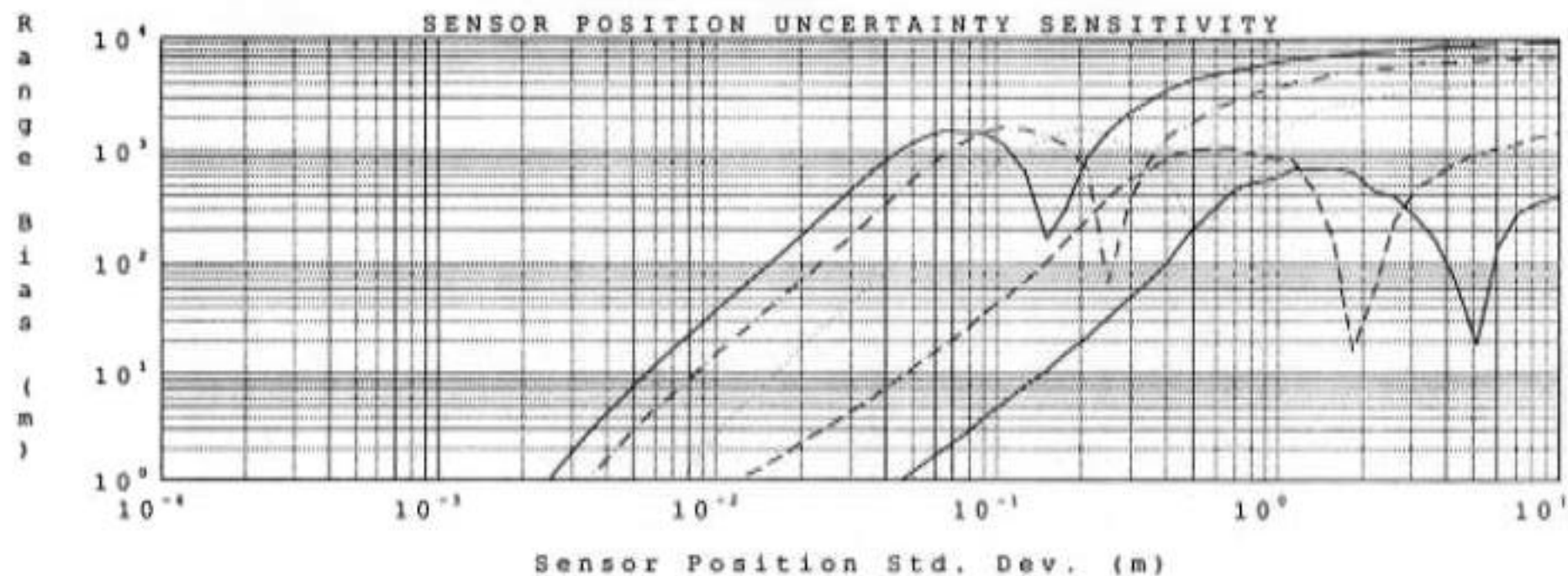


Figure 11: Sensor-Position Uncertainty Sensitivity (Bearing = 90, Sensor Arm = 60 m)

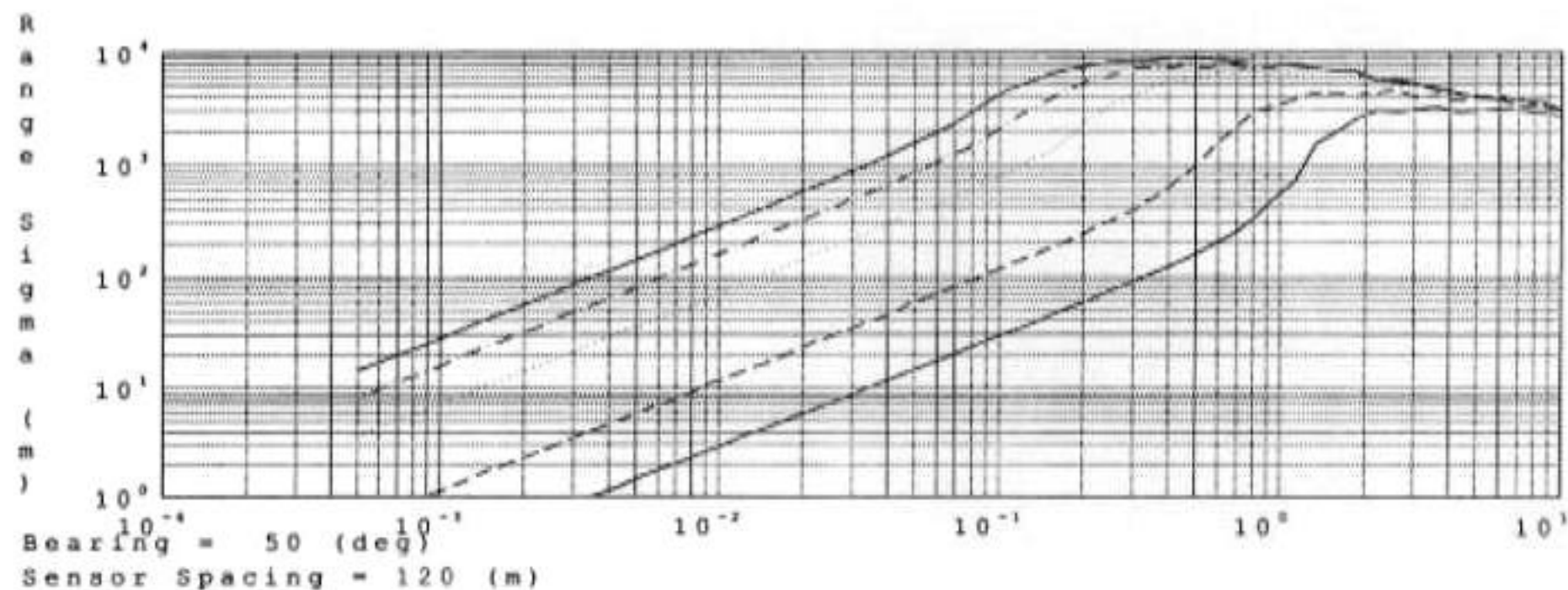
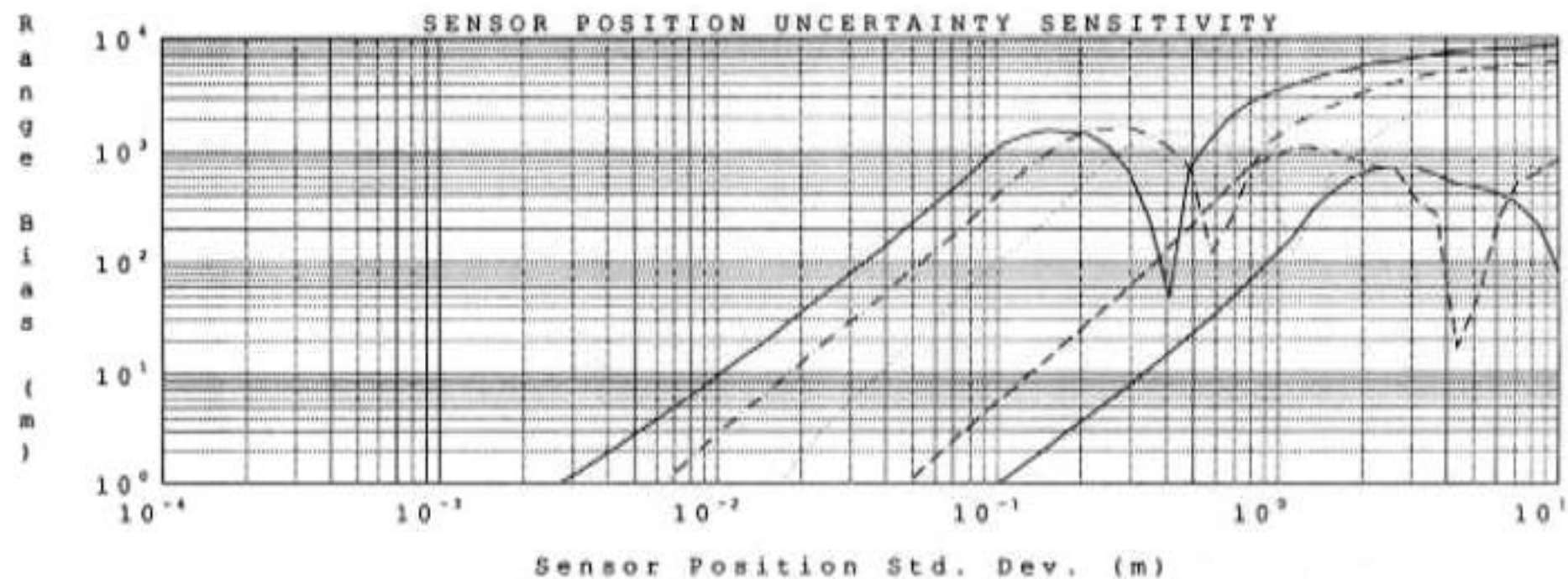


Figure 12: Sensor-Position Uncertainty Sensitivity (Bearing = 50, Sensor Arm = 120 m)

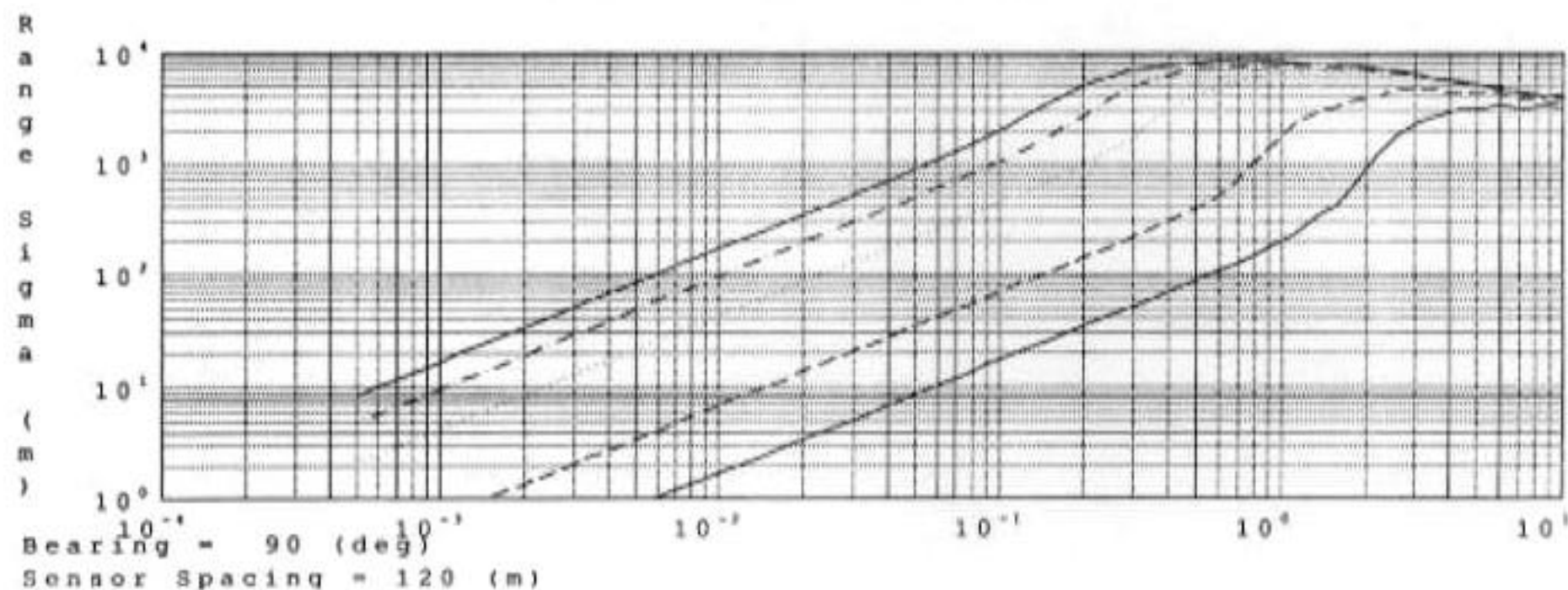
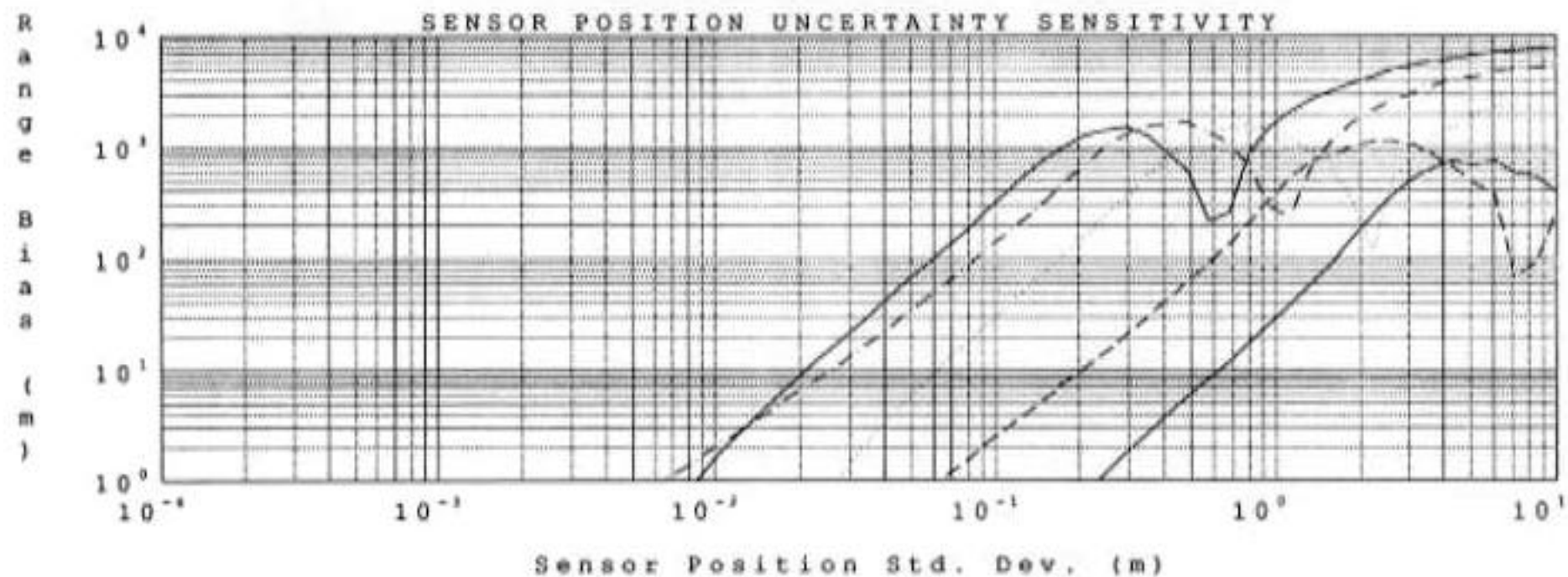


Figure 13: Sensor-Position Uncertainty Sensitivity (Bearing = 90, Sensor Arm = 120 m)

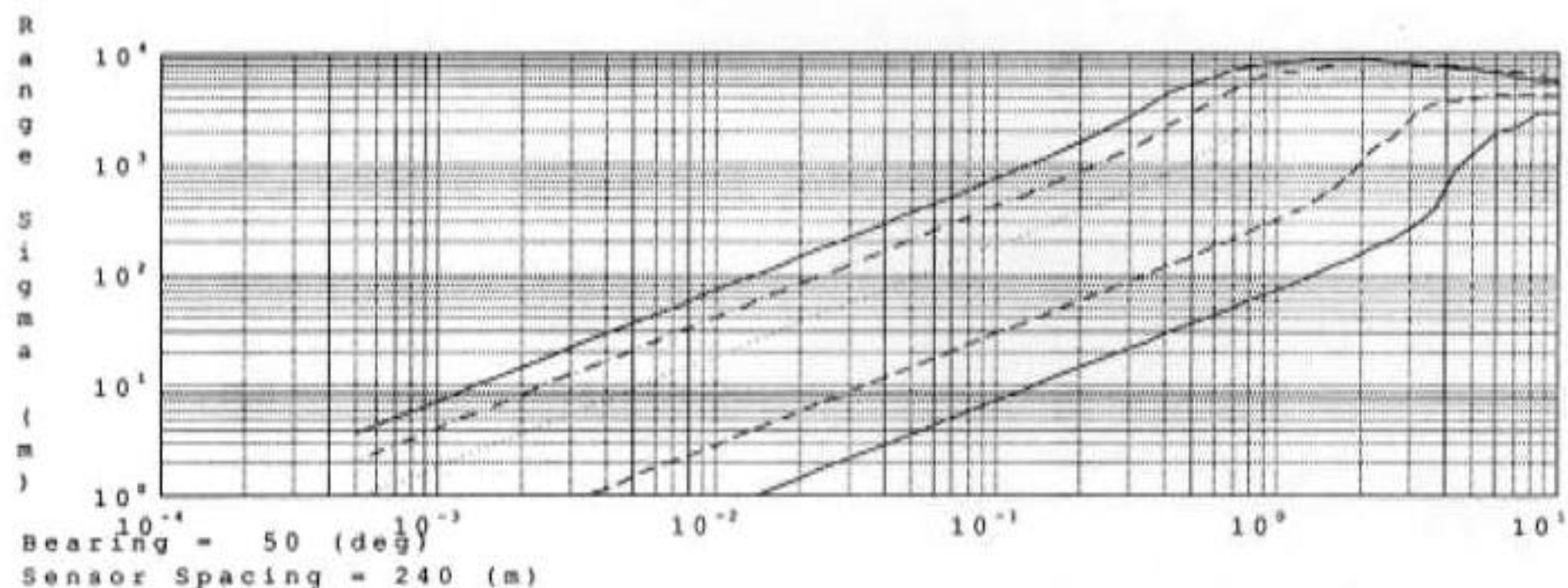
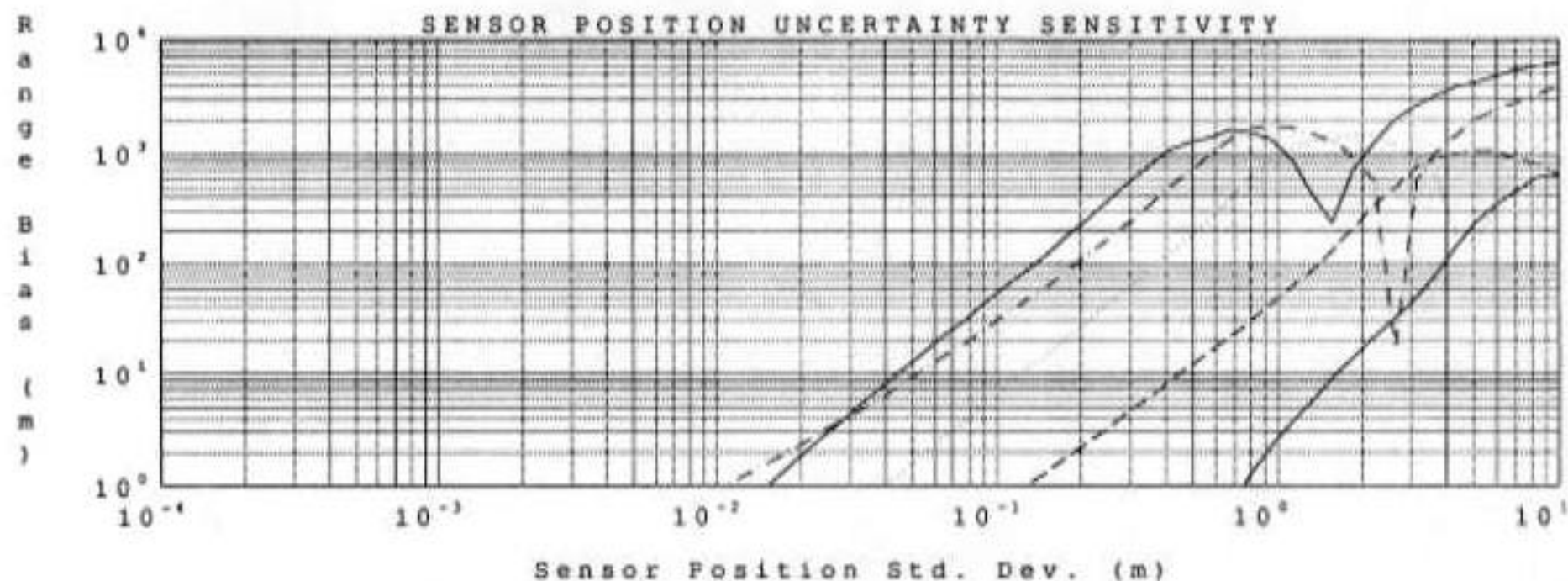


Figure 14: Sensor-Position Uncertainty Sensitivity (Bearing = 50, Sensor Arm = 240 m)

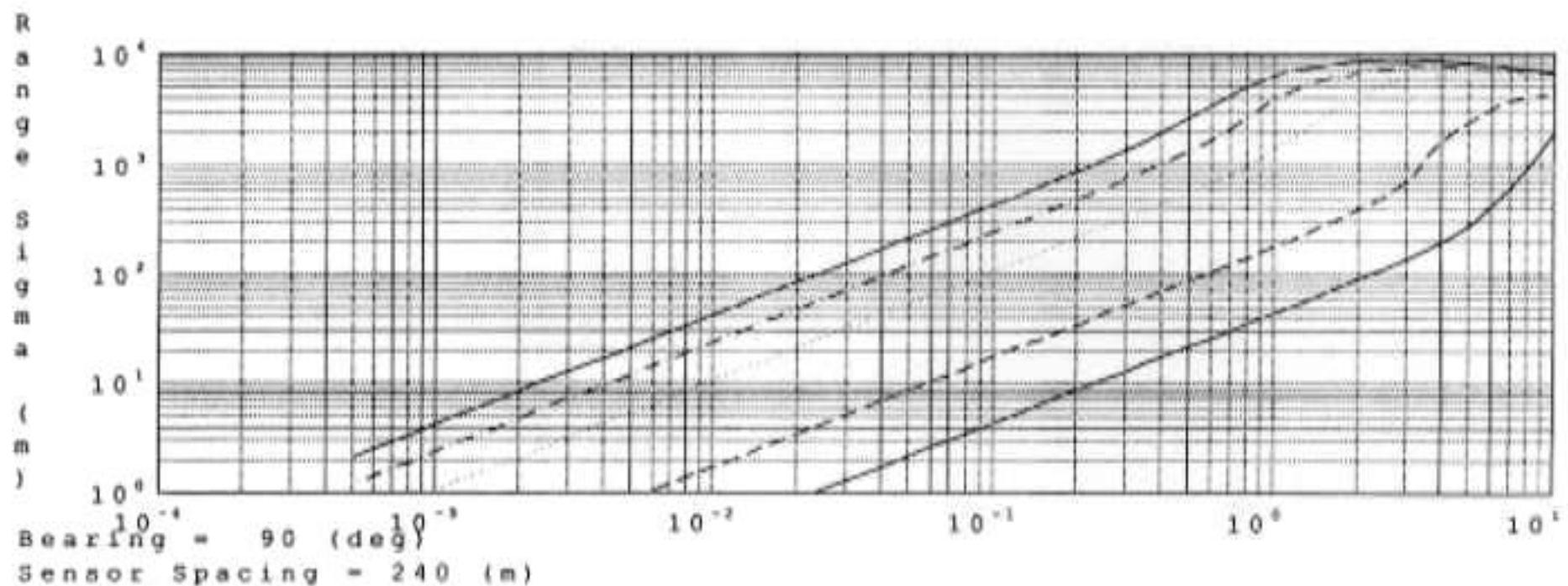
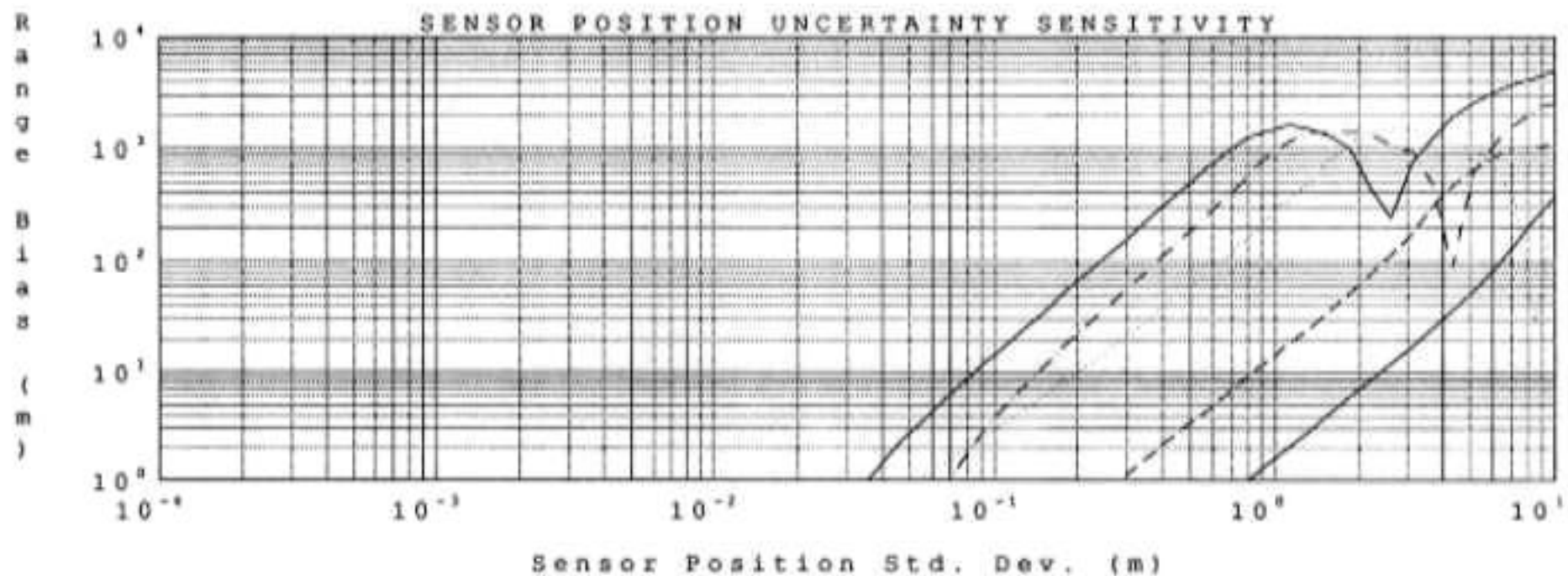


Figure 15: Sensor-Position Uncertainty Sensitivity (Bearing = 90, Sensor Arm = 240 m)

## 5 Time-Delay Estimation

### 5.1 Time-Delay Estimation Algorithms

As noted in Section 3, the wave-front curvature algorithm requires accurate estimates of the time-delay of arrival of the signal at pairs of sensor belonging to a three-sensors linear array. In this section, we now investigate techniques that can be used to estimate the delay of a signal arriving at two spatially separated sensors,  $S_1$  and  $S_2$ . We consider the problem of estimating the TDOA of a Gaussian random noise signal at two sensors when the signal is embedded in independent Gaussian noise at each sensor. The signal, denoted by  $s(t)$ , is not necessarily stationary. The measured signal at  $S_1$  is the sum of  $s(t)$  and a random noise component  $n_1(t)$  while the measured signal at  $S_2$  is the sum of a delayed and scaled version of  $s(t)$  together with random noise  $n_2(t)$ . We have

$$\begin{aligned}x_1(t) &= s(t) + n_1(t) \\x_2(t) &= L s(t - d) + n_2(t)\end{aligned}\tag{5.1}$$

We seek a value of  $d$  which minimizes the difference, in some natural way, between  $x_1$  and  $x_2$ . Two approaches are considered. The first is to choose that value of  $d$  which minimizes the mean-square-difference of the two signals. This is equivalent to locating the peak of the cross-correlation function of the two signals. The second method is to choose a value of  $d$  which maximizes the likelihood that the two signals are the same. While maximum likelihood methods have been used successfully for signals having large time-bandwidth product, we will implement a new formulation for short-duration signals due to Champagne, Eizenman and Pasupathy ([1, 2]) and termed the CEP algorithm. The choice of the appropriate method of time delay estimator will be decided upon using Monte Carlo analysis of repeated trials.

### 5.2 Correlation Method

The correlation method involves computing the cross-correlation function of a sampled version of the continuous signal received at two spatially separated sensors. The location of the peak of this function gives directly the time delay of one signal with respect to another. However, finding the peak of the discrete correlation function of two discrete signals is not optimum as the true delay may not be an integral number of sampling intervals. We choose as a nominal sampling interval, 10kHz. The time-delay sensitivity analysis indicates that good range accuracy puts stringent requirements on allowable time delay errors. Since the

sampling rate of the system under study is larger than that allowable error, we follow the peak determination step with an interpolator to estimate its true peak.

In our experiment, we modelled a discretely sampled signal received at two different sensors but delayed in time from one sensor to another.. We began by constructing an ensemble of 200 time series for each sensor. Each member of the ensemble was the sum of a noise component and a signal component. The noise was generated from a single 400,000 point Gaussian white noise sequence obtained by a call to a MATLAB routine. This whole sequence was divided into two sequences of 200,000 points (one for each sensor) and each of these series was divided into 200 experiments of 1000 points each. This represents 0.1 seconds at a 10,000 kHz sampling rate. The noise power is by definition unity.

We defined the received signal at each sensor by a Gaussian white noise sequence of 0.1 second duration and 100KHz sampling rate. By delaying this signal an integral number of samples of the higher sampling rate, we can model a delay which is not an integral number of low-rate sampling intervals. For 0.1 seconds, we have 10,000 samples. We filtered this sequence through a first-order AR filter defined for three different bandwidths: BW = .5KHz, 1KHz and 2 KHz and for four SNR's: 1 dB, 4 dB, 7 dB, 10 dB. A copy of this signal was made but shifted by 5005 samples at the higher sampling rate. Each signal of the pair was then decimated by a factor of 10 to match the sampling rate of the noise, giving 1000 samples, and added directly to each sensors noise ensemble.

The continuous signal was modeled as a first-order autoregressive random process having as autocorrelation functions and spectra:

$$R_a(\tau) = Pe^{-\alpha|\tau|} \quad (5.2)$$

$$G_a(\omega) = \frac{2\alpha P}{\omega^2 + \alpha^2} \quad (5.3)$$

The correlation time of this signal is defined by

$$\tau_c = \frac{1}{\alpha} \quad (5.4)$$

In order to discretize this process, we define the sampled autocorrelation function

$$R_a(n) = Pe^{-\alpha nT} \quad (5.5)$$

where  $T$  is the sampling interval. The discrete spectral density function comes from taking the z-transform of this function and takes the form

$$G(z) = P \frac{1 + A^2}{1 - A(z + z^{-1}) + A^2} \quad (5.6)$$

where  $A = \exp(-\alpha T)$ . A first-order discrete filter which generates a sequence having this autocorrelation function from a Gaussian white noise process takes the form:

$$H(z) = \frac{B}{1 - Az^{-1}} \quad (5.7)$$

where

$$B = \sqrt{P(1 - A^2)} \quad (5.8)$$

The quantities  $A$  and  $B$  are chosen so as to give the desired bandwidth and signal gain. Table 3 relates the bandwidth and signal correlation time (in samples at 100 kHz).

| BW (Hz) | $\tau_c$ (samples) |
|---------|--------------------|
| 500     | 55.14              |
| 1000    | 27.57              |
| 2000    | 13.80              |

Table 3: Bandwidth and Correlation Time

An estimate of the cross-correlation between two signals over a finite observation time is obtained from the quantity [5]:

$$\hat{R}_{x_1 x_2}(d) = \frac{1}{T - D} \int_t^T x_1(t)x_2(t - d)dt \quad (5.9)$$

or in terms of discrete sampling:

$$\hat{R}_{x_1 x_2}(d) = \frac{1}{N - d} \sum_{n=0}^{N-d-1} x_n x_{n-d} \quad (5.10)$$

In the conventional cross-correlator (CCC), this quantity can be evaluated in the frequency domain using the well-known convolution theorem. Two finite length sequences are converted to the frequency domain using discrete FFT's. The corresponding sequences are then multiplied element-by-element. By taking the inverse FFT of this sequence, we arrive at  $\hat{R}_{x_1 x_2}(d)$ . This gives the cross-correlation on the discrete time grid defined by the sampling of the sensor outputs. A coarse measurement of delay is made by finding the peak of the cross-correlation function. Using an 11-point Lagrange interpolator, a finer peak localization was made by interpolating the cross-correlation function at a mesh of 10 times the sampling rate. The results of the 200 experiments are summarized below in terms of the sample mean delay and sample standard deviation.

| SNR (dB) | Bandwidth (KHz) |                |               |
|----------|-----------------|----------------|---------------|
|          | .5              | 1              | 2             |
| 10       | -0.8<br>(.99)   | 0.2<br>(.77)   | -0.2<br>(.75) |
| 7        | -0.6<br>(1.4)   | 0.1<br>(1.22)  | -0.5<br>(.98) |
| 4        | -0.1<br>(2.40)  | 0.0<br>(1.82)  | 0.5<br>(1.44) |
| 1        | 0.3<br>(3.79)   | -0.9<br>(2.46) | 1.0<br>(2.08) |

Table 4: Mean Delay (and Standard Deviation) (in samples at 100 kHz)

This table shows that even at low SNR's and small bandwidths, fairly good time-delay estimation can be achieved. The largest standard deviation (at 1 dB SNR and 500 Hz BW) is  $38\mu\text{sec}$  which corresponds to a range error of 2000 meters at 10 km but only 20 meters at 2 km (for a 120m arm-length array).

### 5.3 CEP Algorithm

The CEP algorithm was developed to solve the maximum likelihood estimation problem when the duration of the observation interval is small as compared to the signal delay and autocorrelation time. If a signal arriving at two sensors is represented as a vector process by

$$\mathbf{x}(t; d) = \mathbf{s}(t; d) + \mathbf{n}(t), \quad 0 \leq t \leq T \quad (5.11)$$

where

$$\mathbf{s}(t; d) = \begin{bmatrix} s(t) \\ L s(t-d) \end{bmatrix} \quad (5.12)$$

and

$$\mathbf{n}(t) = \begin{bmatrix} n_1(t) \\ n_2(t) \end{bmatrix} \quad (5.13)$$

then one can expand  $\mathbf{x}(t; d)$  in terms of eigen-2-vectors  $\Psi_k(t)$  of the vector correlation matrix  $\mathbf{R}_x(\tau)$ .

$$\mathbf{x}(t; d) = \sum_{k=0}^{\infty} \zeta_k \Psi_k(t) \quad (5.14)$$

where the expansion coefficients are defined by

$$C_k = \int_0^T \Psi_k^T(t) \mathbf{x}(t; d) dt \quad (5.15)$$

and are, necessarily, uncorrelated. Such an expansion is called a Karhunen-Loeve expansion. The eigenvalue equation satisfied by  $\Psi_k(t)$  is

$$\int_0^T \mathbf{R}_s(t-u; d) \Psi_k(u) du = \lambda_k \Psi_k(t), \quad 0 \leq t \leq T \quad (5.16)$$

and the normalization condition

$$\int_0^T \Psi_j^T(t) \Psi_k(t) dt = \delta_{kj} \quad (5.17)$$

The zero correlation between the expansion coefficients lets us define a log-likelihood function of the form

$$\ln \Lambda(\mathbf{x}; d) = \frac{1}{2} \left( \sum_{k=1}^{\infty} \frac{\lambda_k}{1 + \lambda_k} C_k^2 - \sum_{k=1}^{\infty} \ln(1 + \lambda_k) \right) \quad (5.18)$$

and the MLM method is to choose  $d$  as to maximize the log-likelihood function.

CEP show that Equation 5.16 can be reduced to a scalar equation

$$\int_{-d}^T R_s(t-u) \psi_k(u) \rho(u) du = \lambda_k \psi_k(t), \quad -d \leq t \leq T \quad (5.19)$$

with  $\psi_k(t)$  satisfying the normalization condition

$$\int_{-d}^T \psi_j(t) \psi_k(t) \rho(t) dt = \delta_{jk} \quad (5.20)$$

where  $\rho(t)$  is defined by

$$\rho(t) = \begin{cases} 1, & -d < t < 0 \\ 2, & 0 \leq t < T-d \\ 1, & T-d \leq t < T \end{cases} \quad (5.21)$$

and where the vector eigenvectors are related to the scalar ones by

$$\Psi_k(t) = \begin{bmatrix} \psi_k(t) \\ \psi_k(t-d) \end{bmatrix} \quad (5.22)$$

CEP give a procedure for solving Equation 5.19.

Eigenvectors and eigenvalues depend upon the details of the random process defining the signal as well as the delay  $d$  so that the eigenvalue equation must be solved for each proposed value of  $d$ . This makes for a rather complicated processor. Furthermore, an infinite number of terms appear in the log-likelihood function so that some natural cutoff needs to be defined.

An algorithm for calculating the eigenvalues and eigenvectors is present in their paper. Our work is to adapt their algorithm to the statistical process considered here.

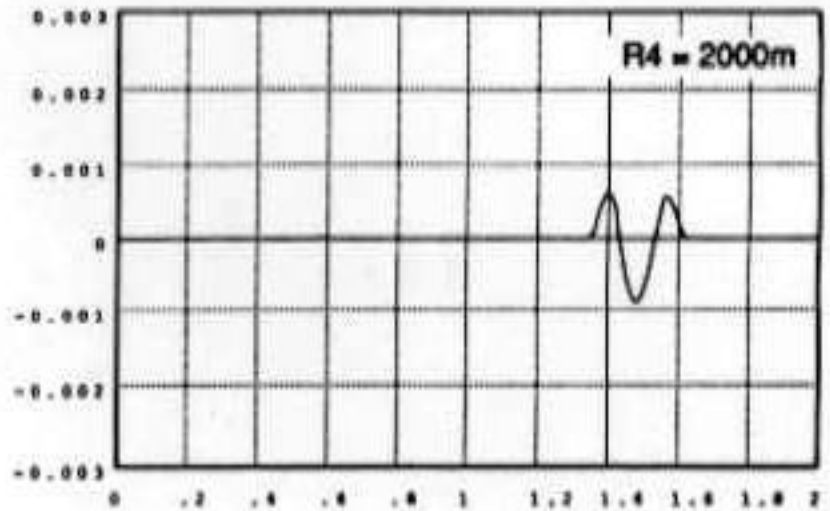
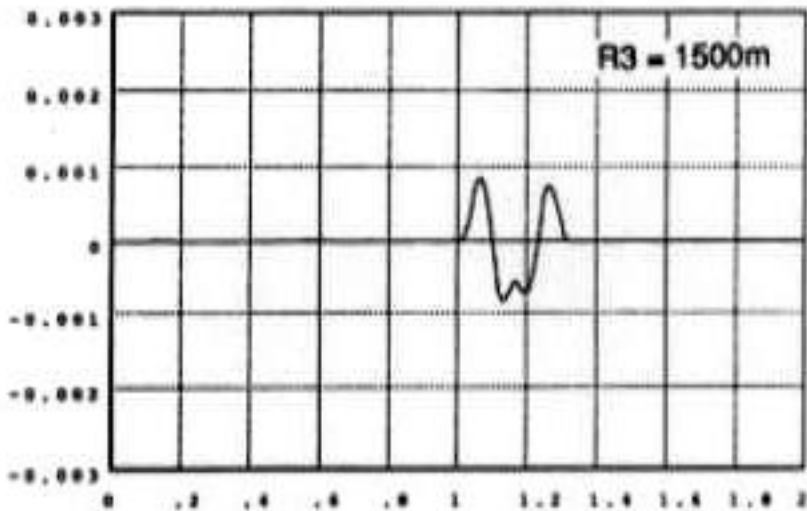
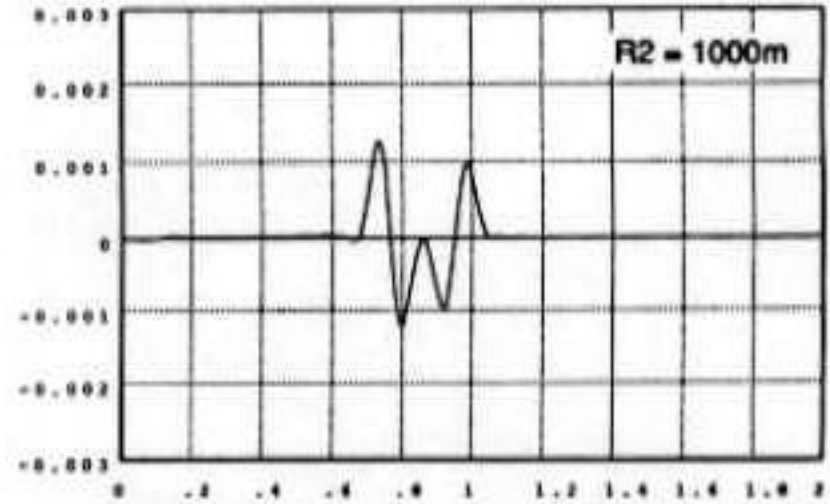
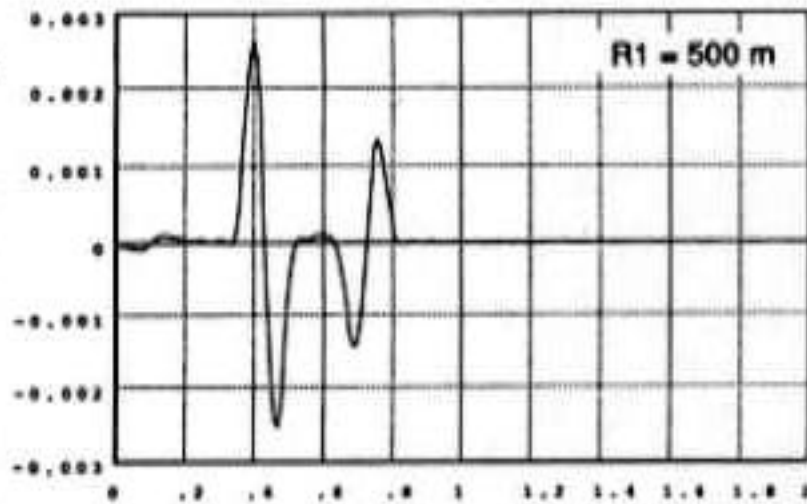
Appendix B describes the calculation (implemented in MATLAB) of the eigenvalues and eigenvectors for a signal whose statistics are the same as that used for the conventional correlator. We have not yet extended the codes to the problem of expanding the signal, calculating the log-likelihood function, and comparing the performance to the conventional cross-correlator. This will be performed in later work.

## 5.4 Full-Wavefield Propagation Models

In order to extend our analysis results to more complicated scenarios such as those involving multipath or dispersive propagation, we have developed the capability at Atlantic Aerospace of using an excellent propagation model, OASES, developed by Henrik Schmidt. The OASES package includes transmission-loss calculation codes and field propagation codes which can take into account a variety of environmental parameters such as surface loss and bottom interaction. OASES is a full-wavefield propagation model in that it numerically integrates the scalar wave-equation. It also contains an interactive display program. Figure 16 below illustrates the use of OASES to show how a signal is propagated to different ranges. It shows a signal propagated from its source to four different ranges. The changing shape is indicative of interference between a direct path component and a surface reflected component.

## Example Time Domain Waveforms Observed at Multiple Sensors

Level



Time (Sec)

Figure 16: Field Propagated by OASES at Multiple Sensors

## 6 Summary

The objectives of this task were to establish acceptable time-delay and sensor position requirements in order to meet reasonable target localization goals and to determine the best algorithm for achieving the time-delay resolution requirements. Our work provides preliminary characterizations of these requirements under a restricted, but reasonable, set of scenarios. The extension of these analyses to more complicated situations is the goal of future work. What will be of great value in future work are the algorithms and software we have developed at AAEC to do short-duration maximum likelihood estimation and generalized asymptotic maximum-likelihood estimation and to study the performance of these and other algorithms in, for example, colored noise, non-Gaussian noise, or in the presence of interference. In addition, using OASES, we have the capability of analyzing the effects of multipath and reverberation. Finally, and perhaps most importantly, we are now capable of using our algorithms and software to analyze real data.

The fundamental analytic conclusions of this report are that localization is difficult at long ranges ( $> 7.5\text{km}$ ) for the bandwidths and sampling frequencies considered and entirely feasible for closer range. This holds true for the conventional cross-correlator but whether this holds for the CEP algorithms remains to be seen. As we saw in Section 3, time delay resolution must be on the order of several microseconds in order to localize a source at 10 km using a 60m array arm-length and about 10 microseconds using an array with a 120m array arm-length. The analysis in Section 5 shows that one can resolve time-delay to about 10 microseconds at 10dB SNR for a 2kHz bandwidth signal and to about 40 microseconds for a .5kHz signal at 1dB SNR. This means that, at short distances, ( $\leq 2\text{km}$ ), these results are more than adequate and at moderate distances, ( $5 - 7.5\text{km}$ ), this resolution is adequate for the 120m arm-length array but not for shorter arrays.

## References

- [1] B. Champagne, M. Eizenman, and S. Pasupathy. "Factorization properties of optimum spacetime processors in nonstationary environments". *IEEE Trans. Acoust., Speech, Signal Processing*, 38(11):1853-1869, November 1990.
- [2] B. Champagne, M. Eizenman, and S. Pasupathy. "Exact maximum likelihood time delay estimation for short observation intervals". *IEEE Trans. Signal Processing*, 39(6):1245-1257, June 1991.
- [3] Wilbur B. Davenport, Jr. and William Root. *Random Signals and Noise*. McGraw-Hill, New York, 1958.
- [4] Joseph C. Hassab. *Underwater Signal and Data Processing*. CRC Press, Boca Raton, 1989.
- [5] C.H. Knapp and G.C. Carter. "The generalized correlation method for estimation of time delay". *IEEE Trans. Acoust., Speech, Signal Processing*, 24(4):320-327, August 1976.

## A Wave-Front Curvature Equations

This appendix gives the derivation of the range and bearing equations from time-delay and acoustic sensor position measurements (recall Figure 2 in Section 2 for a description of the geometry.)

### Notation

- $\vec{R}$  = Slant range vector from Sensor 2 nominal location to source.
- $\hat{R}$  = Unit length slant range vector.
- $R$  = Range of source where  $\vec{R} = R\hat{R}$ .
- $\vec{L}_1$  = Nominal vector from Sensor 2 to Sensor 1.  $L_1 = |\vec{L}_1|$
- $\vec{L}_2$  = Nominal vector from Sensor 2 to Sensor 3.  $L_2 = |\vec{L}_2|$
- $\vec{\delta}_1$  = Sensor 1 offset from nominal location.
- $\vec{\delta}_2$  = Sensor 2 offset from nominal location.
- $\vec{\delta}_3$  = Sensor 3 offset from nominal location.
- $\vec{R}_1$  = Vector from actual Sensor 1 to source.
- $\vec{R}_2$  = Vector from actual Sensor 2 to source.
- $\vec{R}_3$  = Vector from actual Sensor 3 to source.

### Basic Equations

If we assume that the signal was emitted by the source at  $t = 0$ , then we can define the time of arrival (TOA) at the each sensor by  $t_1$ ,  $t_2$  and  $t_3$ . These times are related to the magnitudes of the range vectors by:

$$t_1 = \frac{R_1}{c} \quad (\text{A.1a})$$

$$t_2 = \frac{R_2}{c} \quad (\text{A.1b})$$

$$t_3 = \frac{R_3}{c} \quad (\text{A.1c})$$

What is important for time wave-front curvature ranging is the time delay between pairs of sensors. Actually, for three sensors, we need two pair of delays. We choose

$$\tau_1 = t_1 - t_2 \quad (\text{A.2a})$$

$$\tau_2 = t_2 - t_3 \quad (\text{A.2b})$$

The relations between the nominal ranges from sensor to target are easy to derive from inspection of Figure 1. They are

$$\tilde{R} = \tilde{L}_1 + \tilde{\delta}_1 + \tilde{R}_1 \quad (\text{A.3a})$$

$$\tilde{R} = \tilde{\delta}_2 + \tilde{R}_2 \quad (\text{A.3b})$$

$$\tilde{R} = \tilde{L}_2 + \tilde{\delta}_2 + \tilde{R}_3 \quad (\text{A.3c})$$

The collinearity of the array is expressed by the simple relation

$$\tilde{L}_2 + a\tilde{L}_1 = 0 \quad (\text{A.4})$$

where

$$a = \frac{L_2}{L_1} \quad (\text{A.5})$$

We can use Equations A.1 and A.3 to define the times of arrivals at the nominal reference sensor (Sensor 2).

$$\begin{aligned} c^2 t_1^2 &= |\tilde{R}_1|^2 \\ &= |\tilde{R} - \tilde{L}_1 - \tilde{\delta}_1|^2 \\ &= R^2 + L_1^2 + \delta_1^2 - 2\tilde{R} \cdot \tilde{L}_1 + 2\tilde{L}_1 \cdot \tilde{\delta}_1 - 2\tilde{R} \cdot \tilde{\delta}_1 \end{aligned} \quad (\text{A.6})$$

$$\begin{aligned} c^2 t_2^2 &= |\tilde{R}_2|^2 \\ &= |\tilde{R} - \tilde{\delta}_2|^2 \\ &= R^2 + \delta_2^2 - 2\tilde{R} \cdot \tilde{\delta}_2 \end{aligned} \quad (\text{A.7})$$

$$\begin{aligned} c^2 t_3^2 &= |\tilde{R}_3|^2 \\ &= |\tilde{R} - \tilde{L}_2 - \tilde{\delta}_3|^2 \\ &= R^2 + L_2^2 + \delta_3^2 - 2\tilde{R} \cdot \tilde{L}_2 + 2\tilde{L}_2 \cdot \tilde{\delta}_3 - 2\tilde{R} \cdot \tilde{\delta}_3 \end{aligned} \quad (\text{A.8})$$

Substituting Equations A.6 and A.7 into Equation A.2 gives

$$c\tau_1 = \sqrt{R^2 + L_1^2 + \delta_1^2 - 2\tilde{R} \cdot \tilde{L}_1 + 2\tilde{L}_1 \cdot \tilde{\delta}_1 - 2\tilde{R} \cdot \tilde{\delta}_1} - \sqrt{R^2 + \delta_2^2 - 2\tilde{R} \cdot \tilde{\delta}_2} \quad (\text{A.9})$$

If we assume that the ranges to be measured are much greater than any of the sensor displacements, then the second term on the right-hand-side can be expanded in a Taylor series to terms of first order in  $\delta/R$  and the brought over to the left-hand-side. Thus

$$\begin{aligned}\sqrt{R^2 + \delta_2^2 - 2\vec{R} \cdot \vec{\delta}_2} &= R \sqrt{1 + \left(\frac{\delta_2}{R}\right)^2 - 2\frac{\vec{R} \cdot \vec{\delta}_2}{R}} \\ &\approx R - \vec{R} \cdot \vec{\delta}_2\end{aligned}\quad (\text{A.10})$$

Bringing this term over to the left-hand side of the previous equation and squaring gives

$$(c\tau_1 + R + \vec{R} \cdot \vec{\delta}_2)^2 = R^2 + L_1^2 + \delta_1^2 - 2\vec{R} \cdot \vec{L}_1 + 2\vec{L}_1 \cdot \vec{\delta}_1 - 2\vec{R} \cdot \vec{\delta}_1 \quad (\text{A.11})$$

We can expand the square on the left-hand side to yield

$$(c\tau_1)^2 + R^2 + (\vec{R} \cdot \vec{\delta}_2)^2 + 2c\tau_1 R - 2c\tau_1 \vec{R} \cdot \vec{\delta}_2 - 2\vec{R} \cdot \vec{\delta}_2 = R^2 + L_1^2 + \delta_1^2 - 2\vec{R} \cdot \vec{L}_1 + 2\vec{L}_1 \cdot \vec{\delta}_1 - 2\vec{R} \cdot \vec{\delta}_1 \quad (\text{A.12})$$

Terms involving  $R^2$  cancel out and we collect all terms involving  $R$  on the left-hand side

$$R \left\{ 2c\tau_1 + 2\vec{R} \cdot (\vec{\delta}_1 - \vec{\delta}_2) \right\} + 2\vec{R} \cdot \vec{L}_1 = -(c\tau_1)^2 + L_1^2 + \delta_1^2 + 2\vec{L}_1 \cdot \vec{\delta}_1 + 2c\tau_1 \vec{R} \cdot \vec{\delta}_2 - (\vec{R} \cdot \vec{\delta}_2)^2 \quad (\text{A.13})$$

A similar result for  $\tau_2 = t_2 - t_3$  can be obtained by simply substituting

$$\begin{aligned}s_2 (= -\tau_2) \text{ for } \tau_1 \\ \vec{L}_2 \text{ for } \vec{L}_1 \\ \vec{\delta}_2 \text{ for } \vec{\delta}_1\end{aligned}$$

in Equation A.13. Then we get

$$R \left\{ 2cs_2 + 2\vec{R} \cdot (\vec{\delta}_3 - \vec{\delta}_2) \right\} + 2\vec{R} \cdot \vec{L}_2 = -(cs_2)^2 + L_2^2 + \delta_2^2 + 2\vec{L}_2 \cdot \vec{\delta}_3 + 2cs_2 \vec{R} \cdot \vec{\delta}_2 - (\vec{R} \cdot \vec{\delta}_2)^2 \quad (\text{A.14})$$

## Range Equations

Equations A.13 and A.14 are not quite satisfactory because they contain terms of the form  $\vec{R} \cdot \vec{L}_1$  and  $\vec{R} \cdot \vec{L}_2$  which are related to the source bearing. We would like these equations to be bearing independent, at least to first-order. To achieve this, we divide Equation A.14 by  $a$  and add it to Equation A.13 and then invoke Equation A.4. We get

$$\begin{aligned}R \left\{ 2c \left( \tau_1 + \frac{s_1}{a} \right) + 2\vec{R} \cdot \left[ (\vec{\delta}_1 - \vec{\delta}_2) + \frac{1}{a} (\vec{\delta}_3 - \vec{\delta}_2) \right] \right\} = \\ -c^2 \left( \tau_1^2 + \frac{1}{a^2} s_1^2 \right) + L_1^2 + \frac{1}{a} L_2^2 + \delta_1^2 + \frac{1}{a} \delta_2^2 + \\ 2\vec{L}_1 \cdot (\vec{\delta}_1 - \vec{\delta}_2) + 2c\vec{R} \cdot \vec{\delta}_2 \left( \tau_1 + \frac{1}{a} s_1 \right) - \frac{a+1}{a} (\vec{R} \cdot \vec{\delta}_2)^2\end{aligned}\quad (\text{A.15})$$

To get at our final result, we introduce the following notation

$$N(\tau_1, \tau_2) = \frac{1}{2}L_1^2 \left\{ 1 - \left( \frac{c\tau_1}{L_1} \right)^2 + a \left( 1 - \left( \frac{c\tau_2}{L_2} \right)^2 \right) \right\} \quad (\text{A.16a})$$

$$\Delta\tau = \tau_1 - \frac{1}{a}\tau_2 = \tau_1 + \frac{1}{a}s_2 \quad (\text{A.16b})$$

$$\tilde{\alpha} = \tilde{\delta}_1 - \tilde{\delta}_2 + \frac{1}{a}(\tilde{\delta}_3 - \tilde{\delta}_2) \quad (\text{A.16c})$$

$$\tilde{\beta} = \tilde{\delta}_3 - \tilde{\delta}_1 \quad (\text{A.16d})$$

Then Equation A.15 becomes

$$R = \frac{N(\tau_1, \tau_2) + \tilde{L}_1 \cdot \tilde{\beta} + c\hat{R} \cdot \tilde{\delta}_2 \Delta\tau + \delta_1^2 + \frac{1}{4}\delta_3^2 - \frac{a+1}{2}(\hat{R} \cdot \tilde{\delta}_2)^2}{(2c\Delta\tau + \hat{R} \cdot \tilde{\alpha})} \quad (\text{A.17})$$

Neglecting terms of second order in  $|\delta|^2$  gives

$$R = \frac{N(\tau_1, \tau_2) + \tilde{L}_1 \cdot \tilde{\beta} + c\hat{R} \cdot \tilde{\delta}_2 \Delta\tau}{(2c\Delta\tau + \hat{R} \cdot \tilde{\alpha})} \quad (\text{A.18})$$

At this point it is worth noting that  $R$  is still coupled to the bearing and through the terms in the denominator and numerator involving  $\hat{R}$  but this is only due to the perturbation in the array shape. The term in the denominator is the most critical since the denominator is usually a very small term. When  $\tilde{\alpha} = 0$  this term vanishes.  $\tilde{\alpha}$  is a measure of the deviation of the array from co-linearity. It is zero even if the array is translated or rotated from its nominal position.

## Bearing Equations

To derive the equation for bearing, we start again with Equation A.13 and A.14 which can be re-written as

$$2\tilde{R} \cdot \tilde{L}_1 = -(c\tau_1)^2 + L_1^2 + 2\tilde{L}_1 \cdot \tilde{\delta}_1 + 2c\tau_1 \hat{R} \cdot \tilde{\delta}_2 - R \{ 2c\tau_1 + 2\hat{R} \cdot (\tilde{\delta}_1 - \tilde{\delta}_2) \} \quad (\text{A.19})$$

$$2\tilde{R} \cdot \tilde{L}_2 = -(c\tau_2)^2 + L_2^2 + 2\tilde{L}_2 \cdot \tilde{\delta}_3 - 2c\tau_2 \hat{R} \cdot \tilde{\delta}_2 + R \{ 2c\tau_2 - 2\hat{R} \cdot (\tilde{\delta}_3 - \tilde{\delta}_2) \} \quad (\text{A.20})$$

Dividing Equation A.20 by  $a^2$  and subtracting from Equation A.19 gives

$$2\tilde{R} \cdot \left( \tilde{L}_1 - \frac{1}{a^2}\tilde{L}_2 \right) =$$

$$\begin{aligned}
& - (c\tau_1)^2 + \frac{1}{a^2} (c\tau_2)^2 + \left( L_1^2 - \frac{1}{a^2} L_2^2 \right) + 2 \left( \vec{L}_1 \cdot \vec{\delta}_1 - \frac{1}{a^2} \vec{L}_2 \cdot \vec{\delta}_3 \right) + 2c \left( \tau_1 + \frac{1}{a^2} \tau_2 \right) \dot{R} \cdot \vec{\delta}_2 \\
& - 2R \left\{ c \left( \tau_1 + \frac{1}{a^2} \tau_2 \right) + \dot{R} \cdot (\vec{\delta}_1 - \vec{\delta}_2) - \frac{1}{a^2} \dot{R} \cdot (\vec{\delta}_3 - \vec{\delta}_2) \right\}
\end{aligned} \tag{A.21}$$

Again, we use Equation A.4 to simplify this equation and also introduce the notation

$$2\dot{R} \cdot \vec{L}_1 = RL_1 \cos \Omega \tag{A.22}$$

$$\vec{\gamma} = (\vec{\delta}_1 - \vec{\delta}_2) - \frac{1}{a^2} (\vec{\delta}_3 - \vec{\delta}_2) \tag{A.23}$$

$$\vec{\xi} = \left( \vec{\delta}_1 + \frac{1}{a} \vec{\delta}_3 \right) \tag{A.24}$$

Then, we see that

$$\begin{aligned}
& 2RL_1 \left( \frac{a+1}{a} \right) \cos \Omega = \\
& -2Rc \left( \tau_1 + \frac{1}{a^2} \tau_2 \right) - (c\tau_1)^2 + \frac{1}{a^2} (c\tau_2)^2 + 2\vec{L}_1 \cdot \vec{\xi} \\
& + 2c \left( \tau_1 + \frac{1}{a^2} \tau_2 \right) \dot{R} \cdot \vec{\delta}_2 - 2R\dot{R} \cdot \vec{\gamma}
\end{aligned} \tag{A.25}$$

Dividing this equation by  $R$  and  $L_1$ , we get

$$\begin{aligned}
\cos \Omega = & -\frac{c}{L_1} \frac{a}{a+1} \left( \tau_1 + \frac{1}{a^2} \tau_2 \right) - \frac{c^2}{2RL_1} \frac{a}{a+1} \left( \tau_1^2 - \frac{1}{a^2} \tau_2^2 \right) + \frac{a}{a+1} \frac{\vec{L}_1}{R} \cdot \vec{\xi} \\
& + \frac{c}{RL_1} \frac{a}{a+1} \left( \tau_1 + \frac{1}{a^2} \tau_2 \right) \dot{R} \cdot \vec{\delta}_2 - \frac{a}{a+1} \frac{\dot{R}}{L_1} \cdot \vec{\gamma}
\end{aligned} \tag{A.26}$$

If we now consider the colinear case, then we have the simpler equation

$$\cos \Omega = -\frac{c}{L_1} \frac{a}{a+1} \left( \tau_1 + \frac{1}{a^2} \tau_2 \right) - \frac{c^2}{2RL_1} \frac{a}{a+1} \left( \tau_1^2 - \frac{1}{a^2} \tau_2^2 \right) \tag{A.27}$$

## B CEP Algorithm

There are two basic parts in solving the maximum likelihood equations as posed by Champagne, Eizenman and Pasupathy [2] (from now on referred to as CEP). The first part is to solve the reduced integral equation to yield the eigenstructure of the stationary process  $a(t)$ . The second part is to write the likelihood function using the coefficients of the expansion of the process in terms of these eigenfunctions. This appendix describes the algorithm as it is applied to the particular signal representation given in Section 5. We follow the steps of the algorithm given on pages 1249 – 1250 of [2].

### Solving the Reduced Integral Equation

CEP show that the eigenvector solutions of the reduced integral equation 5.19 take the general form

$$\psi_i(t) = \begin{cases} \psi_{i1}(t) & -d < t < 0 \\ \psi_{i2}(t) & 0 \leq t < T-d \\ \psi_{i3}(t) & T-d \leq t \leq T \end{cases} \quad (\text{B.1})$$

with Laplace transforms

$$\Psi_{i1}(s) = \frac{e^{sd} D^+(s) P_i(s) - Q_i(s)}{\lambda_i D(s^2) - N(s^2)} \quad (\text{B.2a})$$

$$\Psi_{i2}(s) = \frac{Q_i(s) + \epsilon_i e^{-s(T-d)} Q_i(-s)}{\lambda_i D(s^2) - 2N(s^2)} \quad (\text{B.2b})$$

$$\Psi_{i3}(s) = \epsilon_i e^{-s(T-d)} \Psi_{i1}(-s) \quad (\text{B.2c})$$

where  $\epsilon_i = \pm 1$  and  $0 < \lambda_i < 4P/\alpha$ . The solution of the reduced integral equation becomes one of determining the polynomials  $D^+(s)$ ,  $P_i(s)$ ,  $Q_i(s)$ , the eigenvalues  $\lambda_i$ , and  $\epsilon_i$ .

### Canonical Factorization

The first step in finding the eigenvectors is to factor the power spectral density to give  $D^+(s)$ . We have assumed that the signals are short segments of a Gaussian random process which has an autocorrelation function given by

$$R_a(\tau) = P e^{-\alpha|\tau|} \quad (\text{B.3})$$

and its corresponding power spectral density

$$G_a(\omega) = \frac{2\alpha P}{\omega^2 + \alpha^2} = \frac{N(\omega^2)}{D(\omega^2)} \quad (\text{B.4})$$

where we define the numerator and denominator polynomials

$$\begin{aligned} N(\omega^2) &= 2\alpha P \\ D(\omega^2) &= \omega^2 + \alpha^2 \end{aligned} \quad (\text{B.5})$$

In terms of  $\omega^2$ , the degree of the polynomial  $N(\cdot)$  is  $m = 0$  and the degree of the polynomial  $D(\cdot)$  is  $n = 1$  which meets one of the conditions of applicability of the algorithm,  $n > m$ . The canonical factorization of the polynomial  $D(\omega^2)$  is accomplished by continuing it onto the complex  $s$ -plane by letting  $s = j\omega$ , factoring it into its roots, and collecting together all roots on the left-hand side of the complex plane into  $D^+(s)$  and collecting together all roots on the right-hand side of the complex plane into  $D^-(s)$ . We have

$$\begin{aligned} D(s^2) &= -s^2 + \alpha^2 \\ &= (-s + \alpha)(s + \alpha) \end{aligned} \quad (\text{B.6})$$

The zeros of  $D(s^2)$  are at  $s = \pm\alpha$  which gives the canonical factorization

$$D^+(s) = s + \alpha \quad (\text{B.7a})$$

$$D^-(s) = -s + \alpha \quad (\text{B.7b})$$

## Denominator Structure

We now look at the structure of roots of the polynomials in the denominators of equation B.2. Defining, for general  $D(s^2)$  and  $N(s^2)$ ,

$$W_1(s^2) = \lambda D(s^2) - N(s^2) \quad (\text{B.8a})$$

$$W_2(s^2) = \lambda D(s^2) - 2N(s^2) \quad (\text{B.8b})$$

For any value of  $\lambda$ , there are, in general,  $K_1$  zeroes  $\{s_{1k}(\lambda)\}$  of  $W_1(s^2)$  of multiplicity  $m_{1k}$  where  $k = 1, \dots, K_1$ . For  $W_2(s^2)$ , there are, in general,  $K_2$  zeroes  $\{s_{2k}(\lambda)\}$  with multiplicity  $m_{2k}$ ,  $k = 1, \dots, K_2$ .

Since, in our case,  $W_i(\cdot)$  are polynomials of degree one in  $s^2$ , the roots of  $W_1(s^2)$  as a function of  $\lambda$  are given by

$$\begin{aligned} s_{11} &= \sqrt{\alpha^2 - \frac{2\alpha P}{\lambda}} \\ s_{12} &= -\sqrt{\alpha^2 - \frac{2\alpha P}{\lambda}} \end{aligned} \quad (\text{B.9})$$

and for  $W_2(s^2)$ , they are

$$\begin{aligned}s_{21} &= \sqrt{\alpha^2 - \frac{4\alpha P}{\lambda}} \\ s_{22} &= -\sqrt{\alpha^2 - \frac{4\alpha P}{\lambda}}.\end{aligned}\tag{B.10}$$

The roots  $s_{11}$  and  $s_{12}$  are purely imaginary for  $0 < \lambda < 2P/\alpha$  and real for  $2P/\alpha \leq \lambda < 4P/\alpha$ . The roots  $s_{21}$  and  $s_{22}$  are purely imaginary for the entire domain. In our case, every value of  $\lambda$ ,  $W_1(s^2)$  has two distinct roots of multiplicity one. The same is true for  $W_2(s^2)$ . Notice that the roots satisfy the condition that they are symmetrically placed about the real and imaginary axes. For our particular autocorrelation function,  $K_1 = K_2 = 2$  and  $m_{1k} = m_{2k} = 1$  for the  $k$ -th root.

### Numerator Structure

CEP show that the two real-coefficient polynomials,  $P(s)$  and  $Q(s)$ , take the form

$$P(s) = p_0 + p_1 s + \cdots + p_{n-1} s^{n-1} \tag{B.11a}$$

$$Q(s) = q_0 + q_1 s + \cdots + q_{2n-1} s^{2n-1} \tag{B.11b}$$

where, in our example,  $n = 1$ , so that

$$P(s) = p_0 \tag{B.12a}$$

$$Q(s) = q_0 + q_1 s \tag{B.12b}$$

The numerators in equation B.2 then take the form In our case,

$$g_1(s) = e^{sd}(s + \alpha)p_0 - q_0 - q_1 s \tag{B.13a}$$

$$g_2(s, \epsilon) = q_0 + q_1 s + \epsilon e^{-s(\tau-d)}(q_0 - q_1 s) \tag{B.13b}$$

### Existence Condition

CEP show that if polynomials  $P(\cdot)$  and  $Q(\cdot)$  exist such that the singularities in equation B.2 are removable, then the eigenfunctions exist. They equate this condition to the solution of a particular set of simultaneous equations which we now describe.

Each distinct root  $s_{1k}(\lambda)$  gives rise to a set of equations:

$$g_1^{(i)}(s_{1k}(\lambda)) = 0 \tag{B.14}$$

where the superscript  $l$  denotes the derivative of the function with respect to  $s$  and where  $l = 0, \dots, m_{2k} - 1$ . Similarly, each distinct root  $s_{2k}(\lambda)$  gives rise to

$$g_2^{(l)}(s_{2k}(\lambda), \epsilon) = 0 \quad (\text{B.15})$$

where  $l = 0, \dots, m_{2k} - 1$ . In our example, all roots have multiplicity one so that  $l = 0$  and no derivatives are involved. We then get the set of simultaneous equations

$$e^{s_{11}d}(s_{11} + \alpha)p_0 - q_0 - q_1 s_{11} = 0 \quad (\text{B.16a})$$

$$e^{s_{12}d}(s_{12} + \alpha)p_0 - q_0 - q_1 s_{12} = 0 \quad (\text{B.16b})$$

$$q_0 + q_1 s_{21} + \epsilon e^{-s_{21}(T-d)}(q_0 - q_1 s_{21}) = 0 \quad (\text{B.16c})$$

$$q_0 + q_1 s_{22} + \epsilon e^{-s_{22}(T-d)}(q_0 - q_1 s_{22}) = 0 \quad (\text{B.16d})$$

This system of equations is a function of  $\epsilon$  and implicitly a function of  $\lambda$  through the terms  $s_{11}(\lambda)$ ,  $s_{12}(\lambda)$ ,  $s_{21}(\lambda)$  and  $s_{22}(\lambda)$ . At first glance, this is a system of four equations for three unknowns  $p_0, q_0, q_1$ . However, the last two equations are not independent, since, given the symmetry condition  $s_{21}(\lambda) = -s_{22}(\lambda)$ , the third and fourth equations are the same. A similar degeneracy holds for the general case. Thus we have three equations in three unknowns, which we can re-write in matrix form.

$$\begin{pmatrix} e^{s_{11}d}(s_{11} + \alpha) & -1 & -s_{11} \\ e^{s_{12}d}(s_{12} + \alpha) & -1 & -s_{12} \\ 0 & (1 + \epsilon e^{-s_{21}(T-d)}) & s_{22}(1 - \epsilon e^{-s_{22}(T-d)}) \end{pmatrix} \begin{pmatrix} p_0 \\ q_0 \\ q_1 \end{pmatrix} = 0 \quad (\text{B.17})$$

The system has no non-trivial solution unless the determinant of the matrix vanishes which happens for certain discrete values of  $\epsilon$  and  $\lambda$ . It is easy to see this is true when  $s_{11} = 0$  for  $\epsilon = \pm 1$  for then the first two rows are identical. The value of  $\lambda$  in that case is  $2P/\alpha$ . Other values of  $\lambda$  may exist for which the determinant vanish. These may be obtained numerically. An explicit expression for the determinant is

$$\Delta = 2(1 + \epsilon e^{-s_{21}(T-d)}) \left( s_{11}^2 \sinh s_{11}d + \alpha s_{11} \cosh s_{11}d \right) + 2s_{22}(1 - \epsilon e^{-s_{22}(T-d)}) (\alpha \sinh s_{11}d + s_{11} \cosh s_{11}d) \quad (\text{B.18})$$

Figure 17 shows a numerical computation of  $\Delta$  for  $\epsilon = \pm 1$  as a function of  $\lambda$  in the range .001 to 8.0. This was done for the case  $T = .1$ ,  $d = .07$ ,  $P = 1$  and  $\alpha = .5$ . Notice the nulls for both values of  $\epsilon$  at  $\lambda = 2P/\alpha = 4$  and another null at  $\lambda \approx .19$  for  $\epsilon = 1$ . Figures 18 and 19 show  $\Delta$  for the ranges  $10^{-6}$  to  $8 \cdot 10^{-3}$  and for  $10^{-9}$  to  $8 \cdot 10^{-6}$ . These plots show the increasing number of nulls as  $\lambda$  goes to zero. While there are an infinite number of

roots, their contribution to the log-likelihood function diminishes as  $\lambda$  goes to zero. Hence, we need only consider a finite number of them. The exact conditions for truncation need to be determined. We can write the equation  $\Delta = 0$  in a form which allows us to make some statements concerning the location of its root structure. We have for  $\epsilon = +1$

$$s_{22} \tanh \frac{s_{22}}{2}(T-d) = -s_{11} \frac{\tanh s_{11}d + \frac{\pi}{s_{11}}}{\frac{\pi}{s_{11}} \tanh s_{11}d + 1} \quad (\text{B.19})$$

while for  $\epsilon = -1$ , we have

$$s_{22} \coth \frac{s_{22}}{2}(T-d) = -s_{11} \frac{\tanh s_{11}d + \frac{\pi}{s_{11}}}{\frac{\pi}{s_{11}} \tanh s_{11}d + 1} \quad (\text{B.20})$$

Using the fact that the nulls of  $\Delta$  only occur in the domain  $0 < \lambda < 2P/\alpha$ ,  $s_{11}$ ,  $s_{12}$ ,  $s_{21}$  and  $s_{22}$  are purely imaginary and we can write them as  $s_{11} = ia$  and  $s_{21} = ib$ , etc. This leads to, for  $\epsilon = +1$

$$b \tan\left(\frac{b}{2}(T-d)\right) = a \frac{\tan(ad) + \frac{\pi}{a}}{\frac{\pi}{a} \tan(ad) + 1} \quad (\text{B.21})$$

while for  $\epsilon = -1$ , we have

$$b \cot\left(\frac{b}{2}(T-d)\right) = -a \frac{\tan(ad) + \frac{\pi}{a}}{\frac{\pi}{a} \tan(ad) + 1} \quad (\text{B.22})$$

These equations can be solved numerically to locate the roots of the determinant.

When the determinant vanish, one can solve for two of the coefficients ( $p_0, q_0, q_1$ ) in terms of the other one. The remaining coefficient is determined by a normalization condition on the eigenfunction.

## Computation of Residues

After solving for  $p_0, q_0$  and  $q_1$ , we can write

$$\psi_i(t) = c_i \begin{cases} \psi_{i1}(t) = \sum_{k=1}^{K_1} \text{Res}\left(\frac{D^+(s)P_1(s)e^{K_1(t+d)}}{W_1(s^2)}\right)_{s_{1k}(\lambda_i)} & -d < t < 0 \\ \psi_{i2}(t) = \sum_{k=1}^{K_2} \text{Res}\left(\frac{Q_1(s)e^{K_2t}}{W_2(s^2)}\right)_{s_{2k}(\lambda_i)} & 0 \leq t < T-d \end{cases} \quad (\text{B.23})$$

The notation refers to calculating the residue at the roots of  $W_1(\cdot)$  or  $W_2(\cdot)$ . The index  $i$  refers to a particular set of values  $(\lambda_i, c_i)$  which are solutions of equation B.18. For each such

set, we have two roots  $s_{1k}$  and  $s_{2k}$  of equations B.9 and B.10. Defining

$$a_i = \sqrt{\alpha^2 - \frac{2\alpha P}{\lambda_i}} \quad (\text{B.24a})$$

$$b_i = \sqrt{\alpha^2 - \frac{4\alpha P}{\lambda_i}} \quad (\text{B.24b})$$

Then

$$W_1(s^2) = -\lambda(s - ia_i)(s + ia_i) \quad (\text{B.25a})$$

$$W_2(s^2) = -\lambda(s - ib_i)(s + ib_i) \quad (\text{B.25b})$$

and the scalar eigenvectors take the general form

$$\psi_{11}(t) = -\frac{P_2}{\lambda_i} \left( \cos a_i(t+d) + \frac{\alpha}{a} \sin a_i(t+d) \right) \quad (\text{B.26a})$$

$$\psi_{12}(t) = -\frac{1}{\lambda_i} \left( \frac{q_0}{b} \sin bt + q_1 \cos bt \right) \quad (\text{B.26b})$$

and where  $\psi_{13}(t)$  is determined by the symmetry condition:

$$\psi_i(t) = \epsilon_i \psi_i(T-d-t) \quad (\text{B.27})$$

Determinant vs Lambda:  $0.001 < \text{Lambda} < 7.999$

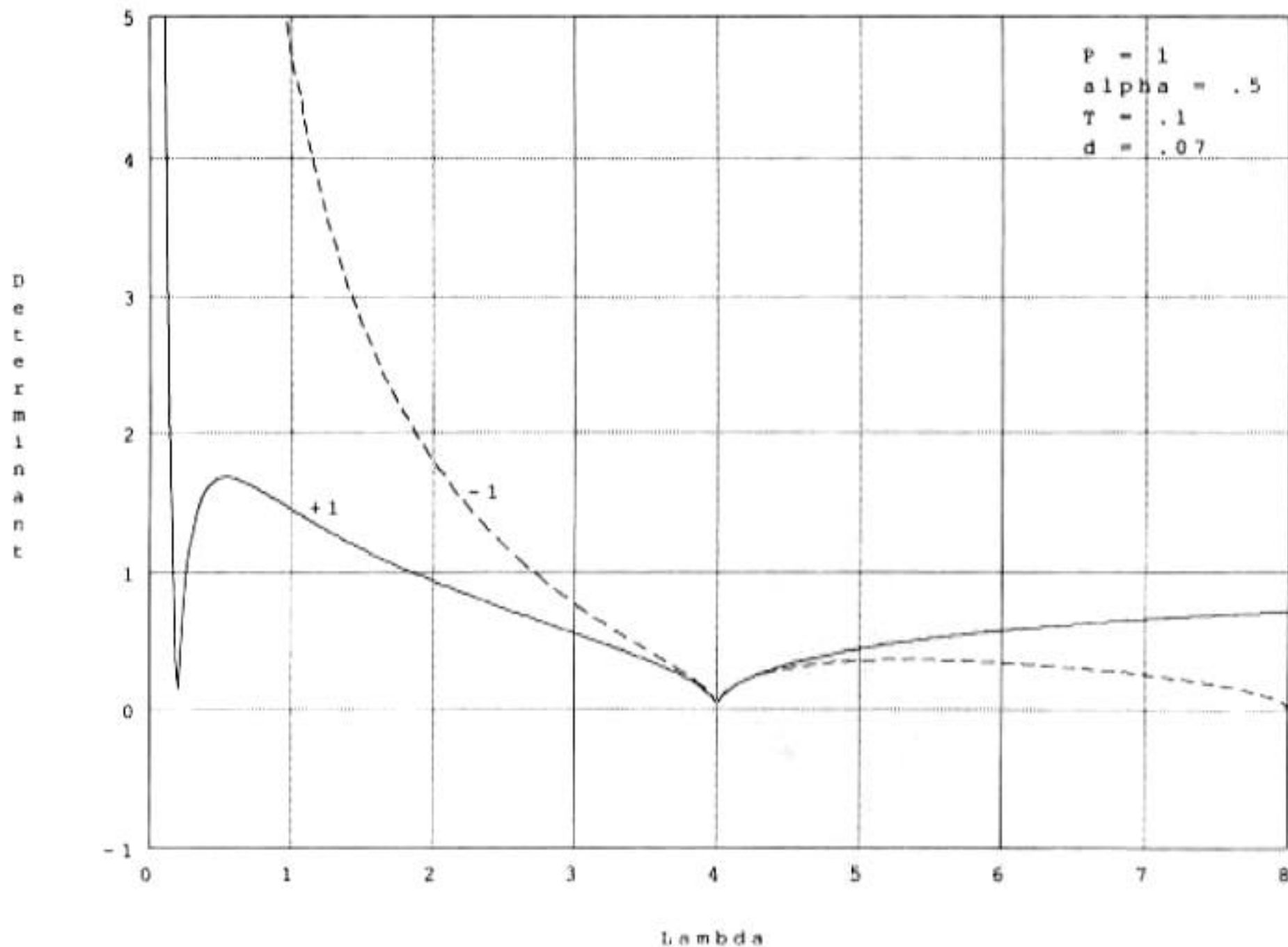


Figure 17: Roots of Determinant:  $.001 < \lambda < 8.0$

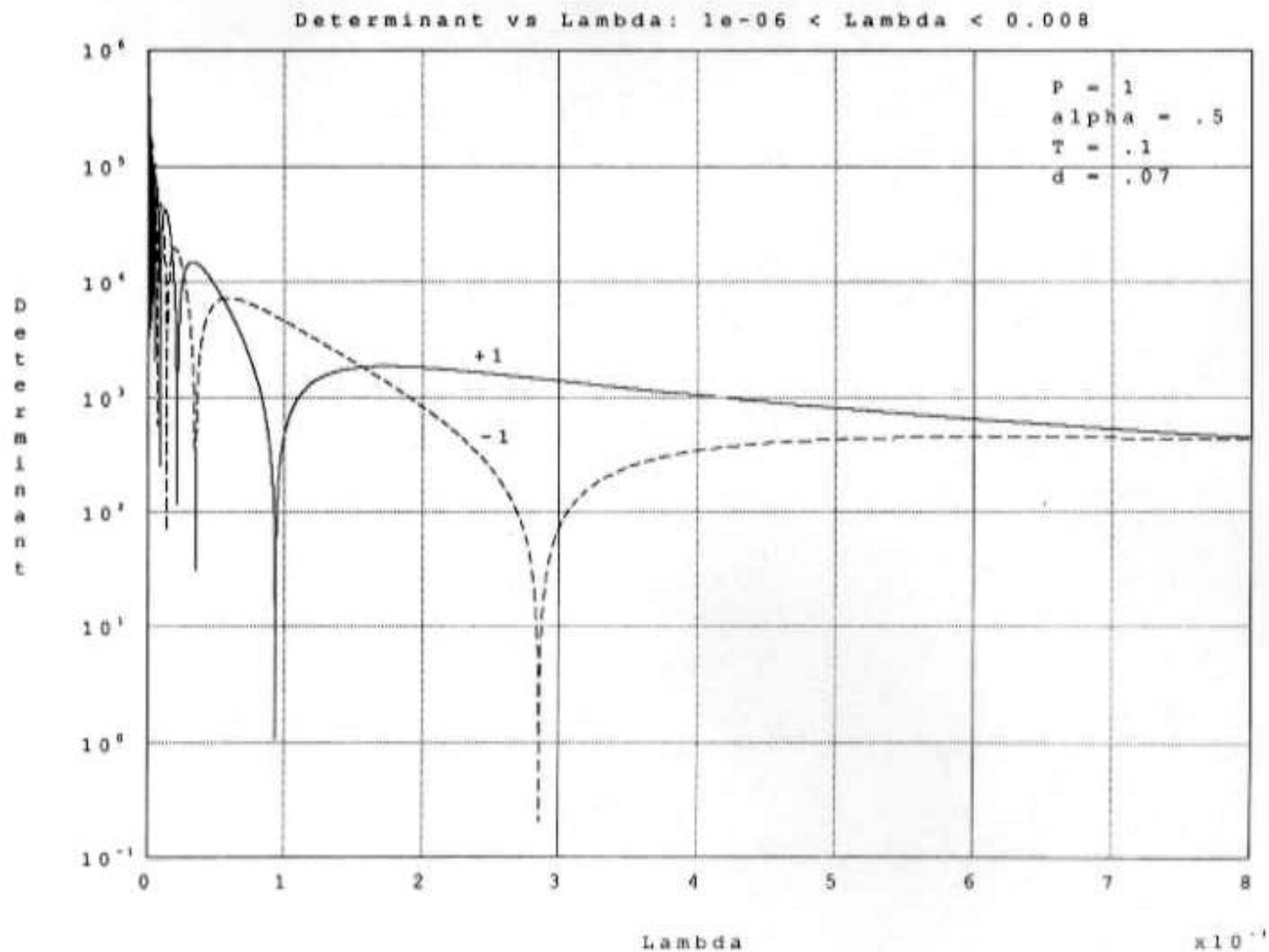


Figure 18: Roots of Determinant:  $10^{-6} < \lambda < 8 \cdot 10^{-3}$

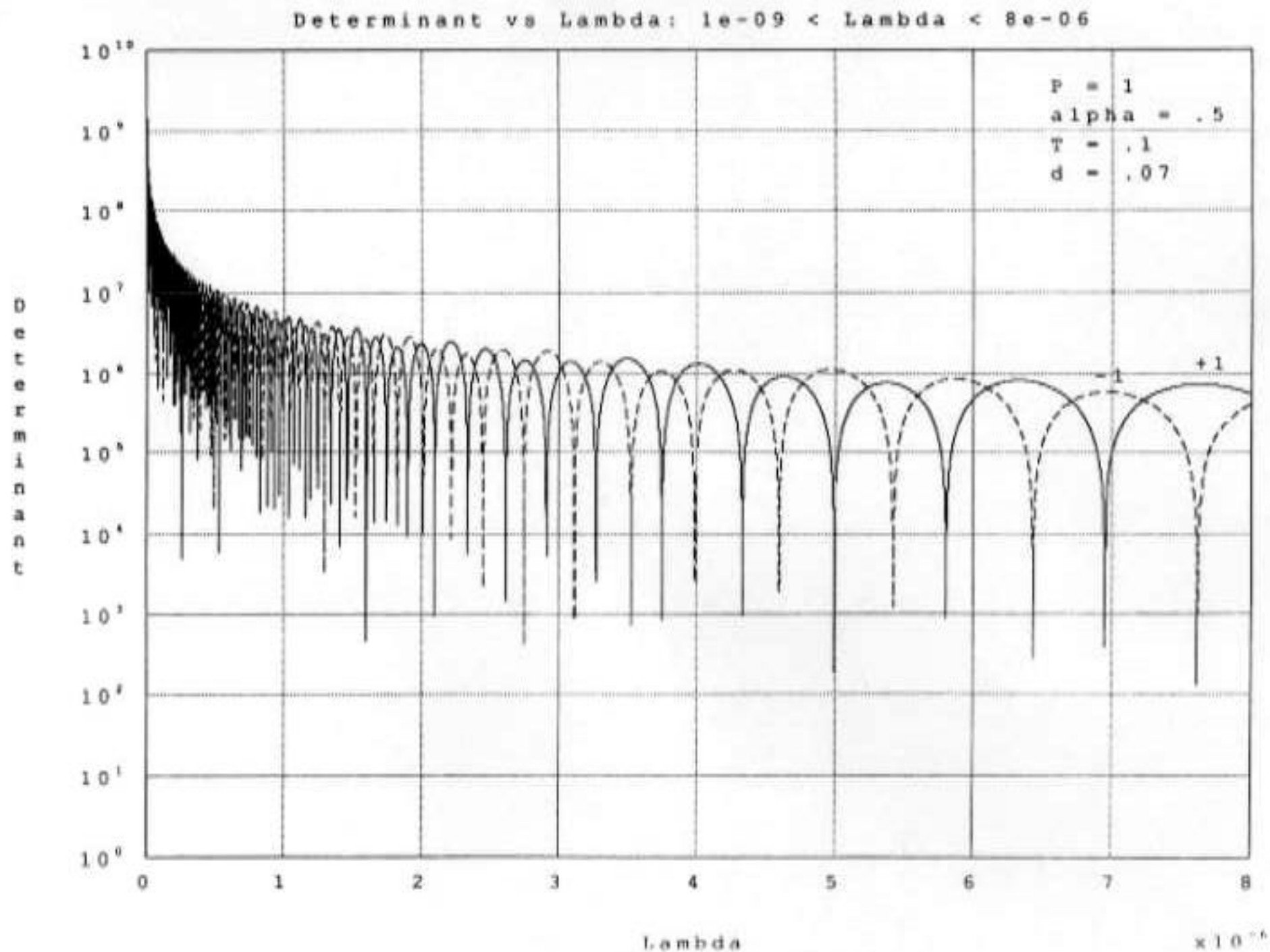


Figure 19: Roots of Determinant:  $10^{-9} < \lambda < 8 \cdot 10^{-6}$

## C Computer Codes

```

% CALC_STAT computes the bias and standard deviation of measured range as a
% function of parameter noise standard deviation.
% The steps of this process are:
%
%     0) Set up outer loop of range for fixed bearing.
%     1) Set up vector array containing position of source as a function
%         of range and bearing.
%     2) Set up 3 vector arrays containing nominal position of receivers.
%
%         The source position is defined by the range and bearing with respect
%         to the nominal center of the array.
%
%     3) set up 3 vector arrays containing known offsets of true receiver
%         positions from nominal positions.
%     4) Calculate true time delays based upon true geometry.
%     5) Calculate measured range and bearing when receiver positions and/or
%         time delays are randomly varied.
%     6) Write the perturbed parameters and perturbed range/bearing to an
%         array.
%     7) Plot the array.
%
% Created 16 July 1991, Ben Rosen, Atlantic Aerospace Electronics Corp.
% Modified 22 July 1991 to loop over multiple ranges, and also processes are
% now vectorized
%
RANGE = [1000 2000 5000 7500 10000];
nr = length(RANGE);
MAX_RANGE = 40000.0 % meters

%bearing = 90; %degrees
bearing = 50; %degrees
DE = 0; %degrees
rbrg = pi/180*bearing;
rDE = pi/180*DE;
%Unit vector to source location
ru = [cos(rbrg)*cos(rDE) sin(rbrg)*cos(rDE) sin(rDE)];

LEN=60;
%LEN=120;
%LEN=240;

% SET UP ARRAY GEOMETRY
xra = [-LEN 0 0]; %meters
xrm = [0 0 0];
xrf = [LEN 0 0];
%
% KNOWN SENSOR POSITION OFFSETS
da = [0 0 0];
dn = [0 0 0];
df = [0 0 0];
%
% Compute true receiver positions
%
xrad = xra + da;
xrnd = xrm + dn;
xrfd = xrf + df;

% COMPUTE GATE VALUE BY SETTING UP MAXIMUM RANGE
xs_gate = MAX_RANGE*ru + xrm; % cartesian location of maximum range
% Compute nominal array lengths
l1 = range(xrf,xrm);
l2 = range(xrm,xra);
a = l2/l1;

c = 1500; %meters/sec
if(SIM == 1)

```

```

rand('normal');

nt = 4000; %Number of samples per experiment

%DEL_SIG = .02; % millisec
%MAX_SIG = 2.0; % millisec
MIN_SIG = .0005; % millisec
MAX_SIG = 10.0; % millisec
NSIG = 60;
%signatau = DEL_SIG:DEL_SIG:MAX_SIG;
signatau = logspace(log10(MIN_SIG),log10(MAX_SIG),NSIG);
signatau=signatau'/1000.0; % transpose and convert to seconds
[rs,cs]=size(signatau);
SIGLOOP = rs; %Number of different variances
%
[taul_gate,tau2_gate] = taus(xs_gate,xrfd,xrmd,xrad,c);
%delta = taul - tau2/a;
GATE = taul_gate - tau2_gate/a

n1 = rand(nt,1);
n2 = rand(nt,1);
n3 = rand(nt,1);

datar_bias = [];
datar_std = [];
datab_bias = [];
datab_std = [];

%OUTER LOOP OVER RANGES

for n=1:nr;
rng = RANGE(n)
clear tau2sav;

% CALCULATE SOURCE POSITION;

xs = rng*ru + xrm;
%
% Compute true time delays
%
[taul,tau2] = taus(xs,xrfd,xrmd,xrad,c);

% Compute approximate bearing using far-field bearing equation.
% Solve for approximate range and bearing using far-field approx.

meas_brg = wfc_bearing(taul,tau2,c,l1,l2,df,dm,da,1000000,DE);
% Is 1000 km far enough?
meas_rng = wfc_range(taul,tau2,c,l1,l2,df,dm,da,meas_brg,0.0);
%
r_bias = [];
r_std_dev = [];
b_bias = [];
b_std_dev = [];

%INNER LOOP OVER NOISE VARIANCES

for isig=1:SIGLOOP;
rand('seed',0);
% Create zero mean noise vectors
noise1 = signatau(isig)*n1;
noise2 = signatau(isig)*n2;
noise3 = signatau(isig)*n3;
%
taulp = taul*ones(nt,1) + noise1 - noise2;
tau2o = tau2*ones(nt,1) + noise2 - noise3;

```

```

delta = tau1p - tau2p/a;
b = wfc_bearing(tau1p,tau2p,c,l1,l2,df,dm,da,abs(meas_rng),DE);
r = wfc_range(tau1p,tau2p,c,l1,l2,df,dm,da,bearing,0.0);
mask = ~(delta-GATE) >= 0.0;
factor2 = 1.0/mean(mask);
factor = sqrt(factor2);
r_bias(isig) = mean(mask.*r)*factor2 - rng;
b_bias(isig) = mean(b) - bearing;
r_std_dev(isig) = std(mask.*r)*factor;
b_std_dev(isig) = std(b);
end;
%END INNER LOOP

datar_bias = [datar_bias r_bias'];
datar_std = [datar_std r_std_dev'];
datab_bias = [datab_bias b_bias'];
datab_std = [datab_std b_std_dev'];
end;
% END OUTER LOOP
%
sigmataau = 1000*sigmataau; %convert to milliseconds.

%CLEANUP
clear n1; clear n2; clear n3;
clear noise1; clear noise2; clear noise3;
clear r_bias; clear r_std_dev; clear b_bias;
clear b_std_dev; clear r; clear b;
clear mask; clear delta;
clear tau1p; clear tau2p;

end;
clf;

lim = [log10(MIN_SIG) log10(MAX_SIG) 0 4];
axis(lim);
%btitle = sprintf('RANGE BIAS vs TIME DELAY UNCERTAINTY');
%stitle = sprintf('RANGE STD. DEV. vs TIME DELAY UNCERTAINTY');
stitle = sprintf('TIME DELAY UNCERTAINTY SENSITIVITY (meters)');
subplot(211);
loglog(sigmataau,abs(datar_bias));
%title(bttitle);
title(stitle);
xlabel('Time Delay Std. Dev. (msec)');
ylabel('Range Bias');
% write out annotations
xs = .01;
ys = .025;
s = sprintf('Bearing = %3.0f (deg)',bearing);
text(xs,ys,s,'sc');
ys = .005;
s = sprintf('Sensor Spacing = %3.0f (m)',l1);
text(xs,ys,s,'sc');

grid;
%meta junk;
subplot(212);
loglog(sigmataau,datar_std);
%title(stitle);
%xlabel('Time Delay Std. Dev. (msec)');
ylabel('Range Sigma');
% write out annotations
% xs = .01;
% ys = .045;
% s = sprintf('Bearing = %3.0f (deg)',bearing);
% text(xs,ys,s,'sc');
% ys = .017;
% s = sprintf('Sensor Spacing = %3.0f (m)',l1);

```

```
%      text(xs,ys,s,'sc');  
grid;  
%meta;  
%print('junk')  
subplot(111);
```

```

function b = wfc_bearing(tau1,tau2,c,l1,l2,df,dm,da,rng,DE)

% function b = wfc_bearing(tau1,tau2,c,l1,l2,df,dm,da,rng,DE)
% Computes bearing to target using wave_front curvature
% equation.
%
% INPUTS
%     tau1 - measured time delay of sensor 1 relative to sensor 2
%     tau2 - measured time delay of sensor 2 relative to sensor 3
%     c    - speed of sound
%     l1   - nominal distance from sensor 1 to 2
%     l2   - nominal distance from sensor 2 to 3
%     rng  - range to target
%
% Created: 1 July 1991, Ben Rosen, Atlantic Aerospace Electronics Corp.
% Last Modified: 1 July 1991, Ben Rosen

a = l2/l1;
ap = a/(a+1);
f1 = c*a/l1/(1+a);
gamma = df - dm - 1/a^2*(da-dm);
xi = df + 1/a*da;
f2 = tau1 + tau2/a^2;
rDE = (pi/180.0)*DE;
cos_brg = -f1*(f2 - 0.5*c*(tau1.*tau1 + tau2.*tau2/a/a)./rng);
b = acos(cos_brg);
[ row,col]=size(b);
cons = sin(rDE)*ones(row,col);
ru = [cos(b).*cos(rDE) sin(b).*cos(rDE)];
ru = [ru cons];
f3 = ap*xi(1)/rng;
f4 = c*ap/rng/l1*f2.*(ru*dm');
f5 = - ap/l1*(ru*gamma');
cos_brg = cos_brg + f3 + f4 + f5;
b = acos(cos_brg)*180/pi;

```

```
function [tau1,tau2] = taus(xs,xr1,xr2,xr3,c)
```

```
% function [tau1,tau2] = taus(xs,xr1,xr2,xr3,c)
```

```
%
```

```
% INPUT
```

```
%     xs - three dim. vector of source position
```

```
%     xr1 - three dim. vector of receiver 1 position
```

```
%     xr2 - three dim. vector of receiver 2 position
```

```
%     xr3 - three dim. vector of receiver 3 position
```

```
%     c  - speed of sound
```

```
%
```

```
% Computes pair of time delays for three sensors. They are
```

```
% (TOA at sensor 1 - TOA at sensor 2) and (TOA at sensor 2 - TOA at  
% sensor 3).
```

```
%
```

```
% Created: 27 June 1991, Ben Rosen, Atlantic Aerospace Electronics Corp
```

```
% Last modified: 1 July 1991, Ben Rosen
```

```
tau1 = time_delay(xs,xr1,xr2,c);
```

```
tau2 = time_delay(xs,xr2,xr3,c);
```

```

function t = travel_time(xs,xr,c)

% function t = travel_time(xs,xr,c)
% INPUTS
%     xs - three dim. vector of source position
%     xr - (N x 3) vector of randomized receiver position
%     c - speed of sound
%
% function td computes the transit time of a signal undergoing straight
% line propagation between a source and receiver
% Created 27 June 1991, Ben Rosen, Atlantic Aerospace Electronics Corp.

[rr,cr]=size(xr);
[rs,cs]=size(xs);
if (rs == 1)
    id = ones(rr,1);
    xsl = id*xs;
    xrl = xr;
end;
if (rr == 1)
    id = ones(rs,1);
    xrl = id*xr;
    xsl = xs;
end;

dx = xrl - xsl;
tp = dx.*dx;
tpl = tp*[1 1 1]';
t = sqrt(tpl)/c;

```

```

% CALC_STAT computes the bias and standard deviation of measured range as a
% function of parameter noise standard deviation.
% The steps of this process are:
%
%   0) Set up outer loop of range for fixed bearing.
%   1) Set up vector array containing position of source as a function
%       of range and bearing.
%   2) Set up 3 vector arrays containing nominal position of receivers.
%
%       The source position is defined by the range and bearing with respect
%       to the nominal center of the array.
%
%   3) set up 3 vector arrays containing known offsets of true receiver
%       positions from nominal positions.
%   4) Calculate true time delays based upon true geometry.
%   5) Calculate measured range and bearing when receiver positions and/or
%       time delays are randomly varied.
%   6) Write the perturbed parameters and perturbed range/bearing to an
%       array.
%   7) Plot the array.
%
% Created 16 July 1991, Ben Rosen, Atlantic Aerospace Electronics Corp.
% Modified 22 July 1991 to loop over multiple ranges, and also processes are
% now vectorized
%
RANGE = [1000 2000 5000 7500 10000];
nr = length(RANGE);
MAX_RANGE = 40000.0 % meters

%bearing = 90; %degrees
bearing = 50; %degrees
DE = 0; %degrees
rbrg = pi/180*bearing;
rDE = pi/180*DE;
%Unit vector to source location
ru = [cos(rbrg)*cos(rDE) sin(rbrg)*cos(rDE) sin(rDE)];

LEN=60;
%LEN=120;
%LEN=240;

% SET UP ARRAY GEOMETRY
xra = [-LEN 0 0]; %meters
xrm = [0 0 0];
xrf = [LEN 0 0];
%
% KNOWN SENSOR POSITION OFFSETS
da = [0 0 0];
dn = [0 0 0];
df = [0 0 0];
%
% Compute true receiver positions
%
xrad = xra + da;
xrmd = xrm + dn;
xrfd = xrf + df;

% COMPUTE GATE VALUE BY SETTING UP MAXIMUM RANGE
xs_gate = MAX_RANGE*ru + xrm; % cartesian location of maximum range
% Compute nominal array lengths
l1 = range(xrf,xrm);
l2 = range(xrm,xra);
a = l2/l1;

c = 1500; %meters/sec
if/SIM == 1)

```

```

rand('normal');

nt = 4000; %Number of samples per experiment

%DEL_SIG = .02; % millisec
%MAX_SIG = 2.0; % millisec
MIN_SIG = .0005; % millisec
MAX_SIG = 10.0; % millisec
NSIG = 60;
%signtau = DEL_SIG:DEL_SIG:MAX_SIG;
signtau = logspace(log10(MIN_SIG),log10(MAX_SIG),NSIG);
signtau=signtau'/1000.0; % transpose and convert to seconds
[rs,cs]=size(signtau);
SIGLOOP = rs; %Number of different variances
%
[taul_gate,tau2_gate] = taus(xs_gate,xrfd,xrmd,xrad,c);
%delta = taul - tau2/a;
GATE = taul_gate - tau2_gate/a

n1 = rand(nt,1);
n2 = rand(nt,1);
n3 = rand(nt,1);

datar_bias = [];
datar_std = [];
datab_bias = [];
datab_std = [];

%OUTER LOOP OVER RANGES

for n=1:nr;
rng = RANGE(n)
clear tau2sav;

% CALCULATE SOURCE POSITION;

xs = rng*ru + xrm;
%
% Compute true time delays
%
[taul,tau2] = taus(xs,xrfd,xrmd,xrad,c);

% Compute approximate bearing using far-field bearing equation.
% Solve for approximate range and bearing using far-field approx.

meas_brg = wfc_bearing(taul,tau2,c,l1,l2,df,dm,da,1000000,DE);
% Is 1000 km far enough?
meas_rng = wfc_range(taul,tau2,c,l1,l2,df,dm,da,meas_brg,0.0);
%
r_bias = [];
r_std_dev = [];
b_bias = [];
b_std_dev = [];

%INNER LOOP OVER NOISE VARIANCES

for isig=1:SIGLOOP;
rand('seed',0);
% Create zero mean noise vectors
noise1 = signtau(isig)*n1;
noise2 = signtau(isig)*n2;
noise3 = signtau(isig)*n3;
%
taulp = taul*ones(nt,1) + noise1 - noise2;
tau2p = tau2*ones(nt,1) + noise3 - noise1;

```

```

delta = tau1p - tau2p/a;
b = wfc_bearing(tau1p,tau2p,c,l1,l2,df,dm,da,abs(meas_rng),DE);
r = wfc_range(tau1p,tau2p,c,l1,l2,df,dm,da,bearing,0.0);
mask = ~(delta-GATE) >= 0.0;
factor2 = 1.0/mean(mask);
factor = sqrt(factor2);
r_bias(isig) = mean(mask.*r)*factor2 - rng;
b_bias(isig) = mean(b) - bearing;
r_std_dev(isig) = std(mask.*r)*factor;
b_std_dev(isig) = std(b);
end;
%END INNER LOOP

datar_bias = [datar_bias r_bias'];
datar_std = [datar_std r_std_dev'];
datab_bias = [datab_bias b_bias'];
datab_std = [datab_std b_std_dev'];
end;
% END OUTER LOOP
%
sigmataau = 1000*sigmataau; %convert to milliseconds.

%CLEANUP
clear n1; clear n2; clear n3;
clear noisel; clear noise2; clear noise3;
clear r_bias; clear r_std_dev; clear b_bias;
clear b_std_dev; clear r; clear b;
clear mask; clear delta;
clear tau1p; clear tau2p;

end;
clg;

lim = [log10(MIN_SIG) log10(MAX_SIG) 0 4];
axis(lim);
%btitle = sprintf('RANGE BIAS vs TIME DELAY UNCERTAINTY');
%stitle = sprintf('RANGE STD. DEV. vs TIME DELAY UNCERTAINTY');
stitle = sprintf('TIME DELAY UNCERTAINTY SENSITIVITY (meters)');
subplot(211);
loglog(sigmataau,abs(datar_bias));
%title(btitle);
title(stitle);
xlabel('Time Delay Std. Dev. (msec)');
ylabel('Range Bias');
% write out annotations
xs = .01;
ys = .025;
s = sprintf('Bearing = %3.0f (deg)',bearing);
text(xs,ys,s,'sc');
ys = .005;
s = sprintf('Sensor Spacing = %3.0f (m)',l1);
text(xs,ys,s,'sc');

grid;
%meta junk;
subplot(212);
loglog(sigmataau,datar_std);
%title(stitle);
xlabel('Time Delay Std. Dev. (msec)');
ylabel('Range Sigma');
% write out annotations
%
xs = .01;
%
ys = .045;
%
s = sprintf('Bearing = %3.0f (deg)',bearing);
%
text(xs,ys,s,'sc');
%
ys = .017;
%
s = sprintf('Sensor Spacing = %3.0f (m)',l1);

```

```
%      text(xs,ys,s,'sc');  
grid;  
%meta;  
%print('junk')  
subplot(111);
```

```

function b = wfc_bearing(tau1,tau2,c,l1,l2,df,dm,da,rng,DE)

% function b = wfc_bearing(tau1,tau2,c,l1,l2,df,dm,da,rng,DE)
% Computes bearing to target using wave_front curvature
% equation.
%
% INPUTS
%     tau1 - measured time delay of sensor 1 relative to sensor 2
%     tau2 - measured time delay of sensor 2 relative to sensor 3
%     c    - speed of sound
%     l1   - nominal distance from sensor 1 to 2
%     l2   - nominal distance from sensor 2 to 3
%     rng  - range to target
%
% Created: 1 July 1991, Ben Rosen, Atlantic Aerospace Electronics Corp.
% Last Modified: 1 July 1991, Ben Rosen

a = l2/l1;
ap = a/(a+1);
f1 = c*a/l1/(1+a);
gamma = df - dm - 1/a^2*(da-dm);
xi = df + 1/a*da;
f2 = tau1 + tau2/a^2;
rDE = (pi/180.0)*DE;
cos_brg = -f1*(f2 - 0.5*c*(tau1.*tau1 + tau2.*tau2/a/a)./rng);
b = acos(cos_brg);
[ row,col]=size(b);
cons = sin(rDE)*ones(row,col);
ru = [cos(b).*cos(rDE) sin(b).*cos(rDE)];
ru = [ru cons];
f3 = ap*xi(1)/rng;
f4 = c*ap/rng/l1*f2.*(ru*dm');
f5 = - ap/l1*(ru*gamma');
cos_brg = cos_brg + f3 + f4 + f5;
b = acos(cos_brg)*180/pi;

```

```

function [taul,tau2] = taus(xs,xr1,xr2,xr3,c)
% function [taul,tau2] = taus(xs,xr1,xr2,xr3,c)
%
% INPUT
%      xs - three dim. vector of source position
%      xr1 - three dim. vector of receiver 1 position
%      xr2 - three dim. vector of receiver 2 position
%      xr3 - three dim. vector of receiver 3 position
%      c - speed of sound
%
% Computes pair of time delays for three sensors. They are
% (TOA at sensor 1 - TOA at sensor 2) and (TOA at sensor 2 - TOA at
% sensor 3).
%
% Created: 27 June 1991, Ben Rosen, Atlantic Aerospace Electronics Corp
% Last modified: 1 July 1991, Ben Rosen

taul = time_delay(xs,xr1,xr2,c);
tau2 = time_delay(xs,xr2,xr3,c);

```

```
function t = travel_time(xs,xr,c)
```

```
% function t = travel_time(xs,xr,c)
```

```
% INPUTS
```

```
%     xs - three dim. vector of source position
```

```
%     xr - (N x 3) vector of randomized receiver position
```

```
%     c - speed of sound
```

```
% function td computes the transit time of a signal undergoing straight  
% line propagation between a source and receiver
```

```
% Created 27 June 1991, Ben Rosen, Atlantic Aerospace Electronics Corp.
```

```
[rr,cr]=size(xr);
```

```
[rs,cs]=size(xs);
```

```
if (rs == 1)
```

```
    id = ones(rr,1);
```

```
    xsl = id*xs;
```

```
    xrl = xr;
```

```
end;
```

```
if (rr == 1)
```

```
    id = ones(rs,1);
```

```
    xrl = id*xr;
```

```
    xsl = xs;
```

```
end;
```

```
dx = xrl - xsl;
```

```
tp = dx.*dx;
```

```
tpl = tp*[1 1 1]';
```

```
t = sqrt(tpl)/c;
```

```

% CALC_STAT computes the bias and standard deviation of measured range as a
% function of parameter noise standard deviation.
% The steps of this process are:
%
%     0) Set up outer loop of range for fixed bearing.
%     1) Set up vector array containing position of source as a function
%         of range and bearing.
%     2) Set up 3 vector arrays containing nominal position of receivers.
%
%         The source position is defined by the range and bearing with respect
%         to the nominal center of the array.
%
%     3) set up 3 vector arrays containing known offsets of true receiver
%         positions from nominal positions.
%     4) Calculate true time delays based upon true geometry.
%     5) Calculate measured range and bearing when receiver positions and/or
%         time delays are randomly varied.
%     6) Write the perturbed parameters and perturbed range/bearing to an
%         array.
%     7) Plot the array.
%
% Created 16 July 1991, Ben Rosen, Atlantic Aerospace Electronics Corp.
% Modified 22 July 1991 to loop over multiple ranges, and also processes are
% now vectorized
%
RANGE = [1000 2000 5000 7500 10000];
nr = length(RANGE);
MAX_RANGE = 40000.0 % meters

%bearing = 90; %degrees
bearing = 50; %degrees
DE = 0; %degrees
rbrg = pi/180*bearing;
rDE = pi/180*DE;
%Unit vector to source location
ru = [cos(rbrg)*cos(rDE) sin(rbrg)*cos(rDE) sin(rDE)];

LEN=60;
%LEN=120;
%LEN=240;

% SET UP ARRAY GEOMETRY
xra = [-LEN 0 0]; %meters
xrm = [0 0 0];
xrf = [LEN 0 0];
%
% KNOWN SENSOR POSITION OFFSETS
da = [0 0 0];
dm = [0 0 0];
df = [0 0 0];
%
% Compute true receiver positions
%
xrad = xra + da;
xrmd = xrm + dm;
xrfd = xrf + df;

% COMPUTE GATE VALUE BY SETTING UP MAXIMUM RANGE
xs_gate = MAX_RANGE*ru + xrm; % cartesian location of maximum range
% Compute nominal array lengths
l1 = range(xrf,xrm);
l2 = range(xrm,xra);
a = l2/l1;

c = 1500; %meters/sec
if(SIM == 1)

```

```

rand('normal');

nt = 4000; %Number of samples per experiment

%DEL_SIG = .02; % millisec
%MAX_SIG = 2.0; % millisec
MIN_SIG = .0005; % millisec
MAX_SIG = 10.0; % millisec
NSIG = 60;
%sigmat = DEL_SIG:DEL_SIG:MAX_SIG;
sigmat = logspace(log10(MIN_SIG),log10(MAX_SIG),NSIG);
sigmat=sigmat'/1000.0; % transpose and convert to seconds
[rs,cs]=size(sigmat);
SIGLOOP = rs; %Number of different variances
%
[tau1_gate,tau2_gate] = taus(xs_gate,xrfd,xrmd,xrad,c);
%delta = tau1 - tau2/a;
GATE = tau1_gate - tau2_gate/a

n1 = rand(nt,1);
n2 = rand(nt,1);
n3 = rand(nt,1);

datar_bias = [];
datar_std = [];
datab_bias = [];
datab_std = [];

%OUTER LOOP OVER RANGES
for n=1:nr;
rng = RANGE(n)
clear tau2sav;

% CALCULATE SOURCE POSITION;
xs = rng*ru + xrn;
%
% Compute true time delays
%
[tau1,tau2] = taus(xs,xrfd,xrmd,xrad,c);

% Compute approximate bearing using far-field bearing equation.
% Solve for approximate range and bearing using far-field approx.

meas_brg = wfc_bearing(tau1,tau2,c,l1,l2,df,dm,da,1000000,DE);
% Is 1000 km far enough?
meas_rng = wfc_range(tau1,tau2,c,l1,l2,df,dm,da,meas_brg,0.0);
%
r_bias = [];
r_std_dev = [];
b_bias = [];
b_std_dev = [];

%INNER LOOP OVER NOISE VARIANCES
for isig=1:SIGLOOP;
rand('seed',0);
% Create zero mean noise vectors
noise1 = sigmat(isig)*n1;
noise2 = sigmat(isig)*n2;
noise3 = sigmat(isig)*n3;
%
taulp = tau1*ones(nt,1) + noise1 - noise2;
tau2p = tau2*ones(nt,1) + noise2 - noise3;

```

```

delta = tau1p - tau2p/a;
b = wfc_bearing(tau1p,tau2p,c,11,12,df,dn,da,abs(meas_rng),DE);
r = wfc_range(tau1p,tau2p,c,11,12,df,dn,da,bearing,0.0);
mask = ((delta-GATE) >= 0.0);
factor2 = 1.0/mean(mask);
factor = sqrt(factor2);
r_bias(isig) = mean(mask.*r)*factor2 - rng;
b_bias(isig) = mean(b) - bearing;
r_std_dev(isig) = std(mask.*r)*factor;
b_std_dev(isig) = std(b);
end;
%END INNER LOOP

datar_bias = [datar_bias r_bias'];
datar_std = [datar_std r_std_dev'];
datab_bias = [datab_bias b_bias'];
datab_std = [datab_std b_std_dev'];
end;
% END OUTER LOOP
%
signatau = 1000*sigmatau; %convert to milliseconds.

%CLEANUP
clear n1; clear n2; clear n3;
clear noise1; clear noise2; clear noise3;
clear r_bias; clear r_std_dev; clear b_bias;
clear b_std_dev; clear r; clear b;
clear mask; clear delta;
clear tau1p; clear tau2p;

end;
clf;

lim = [log10(MIN_SIG) log10(MAX_SIG) 0 4];
axis(lim);
%btitle = sprintf('RANGE BIAS vs TIME DELAY UNCERTAINTY');
%stitle = sprintf('RANGE STD. DEV. vs TIME DELAY UNCERTAINTY');
stitle = sprintf('TIME DELAY UNCERTAINTY SENSITIVITY (meters)');
subplot(211);
loglog(signatau,abs(datar_bias));
%title(btitle);
title(stitle);
xlabel('Time Delay Std. Dev. (msec)');
ylabel('Range Bias');
% write out annotations
xs = .01;
ys = .025;
s = sprintf('Bearing = %3.0f (deg)',bearing);
text(xs,ys,s,'sc');
ys = .005;
s = sprintf('Sensor Spacing = %3.0f (m)',11);
text(xs,ys,s,'sc');

grid;
%meta junk;
subplot(212);
loglog(signatau,datar_std);
%title(stitle);
%xlabel('Time Delay Std. Dev. (msec)');
ylabel('Range Sigma');
% write out annotations
% xs = .01;
% ys = .045;
% s = sprintf('Bearing = %3.0f (deg)',bearing);
% text(xs,ys,s,'sc');
% ys = .017;
% s = sprintf('Sensor Spacing = %3.0f (m)',11);

```

```
%      text(xs,ys,s,'sc');  
grid;  
%meta;  
%print('junk')  
subplot(111);
```

```

function r = wfc_range(tau1,tau2,c,l1,l2,df,dm,da,bearing,DE)

% function r = wfc_range(tau1,tau2,c,l1,l2,df,dm,da,bearing,DE)
% INPUTS
%     tau1 - time delay between forward sensor and middle sensor
%     tau2 - time delay between middle sensor and aft sensor
%     l1 - distance to forward sensor from middle sensor
%     l2 - distance to aft sensor from middle sensor
%     c - speed of sound
%     df - offset in position of forward sensor
%     dm - offset in position of middle sensor
%     da - offset in position of aft sensor
%     bearing - source bearing angle (deg)
%     DE- source depression/elevation angle (deg)
%
% Calculates Range to Target Using Wavefront Curvature Algorithm
% Created 27 June 1991, Ben Rosen, Atlantic Aerospace Electronics Corp

a = l2/l1;
n = 0.5*l1^2*(1.0 + a - (c*tau1/l1).^2 - a*(c*tau2/l2).^2);
dt = tau1 - tau2/a;
beta = df - da;
lbeta = l1*beta(1);
alpha = df - dm + (da - dm)/a;
rbear = (pi/180.0)*bearing;
rDE = (pi/180.0)*DE;
[ row,col]=size(bearing);
cons = sin(rDE)*ones(row,col);
ru = [cos(rbear).*cos(rDE) sin(rbear).*cos(rDE)];
ru = [ru cons];
rm = ru*dm';
ra = ru*alpha';
rd = ru*dm';
rm2 = -0.5*(a+1)/a*(rm^2);% + 0.5*(a+1)/a*(dm*dm');
d1 = 0.5*df*df';
d3 = 0.5*da*da'/a;
r = (n + lbeta + c*dt*rm + d1 + d3 + rm2)./(c*dt + ra);

```

```

function b = wfc_bearing(tau1,tau2,c,l1,l2,df,dm,da,rng,DE)

% function b = wfc_bearing(tau1,tau2,c,l1,l2,df,dm,da,rng,DE)
% Computes bearing to target using wave_front curvature
% equation.
%
% INPUTS
%     tau1 - measured time delay of sensor 1 relative to sensor 2
%     tau2 - measured time delay of sensor 2 relative to sensor 3
%     c    - speed of sound
%     l1    - nominal distance from sensor 1 to 2
%     l2    - nominal distance from sensor 2 to 3
%     rng   - range to target
%
% Created: 1 July 1991, Ben Rosen, Atlantic Aerospace Electronics Corp.
% Last Modified: 1 July 1991, Ben Rosen

a = l2/l1;
ap = a/(a+1);
f1 = c*a/l1/(1+a);
gamma = df - dm - 1/a^2*(da-dm);
xi = df + 1/a*da;
f2 = tau1 + tau2/a^2;
rDE = (pi/180.0)*DE;
cos_brg = -f1*(f2 - 0.5*c*(tau1.*tau1 + tau2.*tau2/a/a)./rng);
b = acos(cos_brg);
[row,col]=size(b);
cons = sin(rDE)*ones(row,col);
ru = [cos(b).*cos(rDE) sin(b).*cos(rDE)];
ru = [ru cons];
f3 = ap*xi(1)/rng;
f4 = c*ap/rng/l1*f2.*(ru*dm');
f5 = - ap/l1*(ru*gamma');
cos_brg = cos_brg + f3 + f4 + f5;
b = acos(cos_brg)*180/pi;

```

```

function [tau1,tau2] = taus(xs,xr1,xr2,xr3,c)

% function [tau1,tau2] = taus(xs,xr1,xr2,xr3,c)
%
% INPUT
%     xs - three dim. vector of source position
%     xr1 - three dim. vector of receiver 1 position
%     xr2 - three dim. vector of receiver 2 position
%     xr3 - three dim. vector of receiver 3 position
%     c - speed of sound
%
% Computes pair of time delays for three sensors. They are
% (TOA at sensor 1 - TOA at sensor 2) and (TOA at sensor 2 - TOA at
% sensor 3).
%
% Created: 27 June 1991, Ben Rosen, Atlantic Aerospace Electronics Corp
% Last modified: 1 July 1991, Ben Rosen

tau1 = time_delay(xs,xr1,xr2,c);
tau2 = time_delay(xs,xr2,xr3,c);

```

```

function t = travel_time(xs,xr,c)

% function t = travel_time(xs,xr,c)
% INPUTS
%     xs - three dim. vector of source position
%     xr - (N x 3) vector of randomized receiver position
%     c - speed of sound
%
% function td computes the transit time of a signal undergoing straight
% line propagation between a source and receiver
% Created 27 June 1991, Ben Rosen, Atlantic Aerospace Electronics Corp.

[rr,cr]=size(xr);
[rs,cs]=size(xs);
if (rs == 1)
    id = ones(rr,1);
    xsl = id*xs;
    xrl = xr;
    end;
if (rr == 1)
    id = ones(rs,1);
    xrl = id*xr;
    xsl = xs;
    end;

dx = xrl - xsl;
tp = dx.*dx;
tpl = tp*[1 1 1]';
t = sqrt(tpl)/c;

```

```

function td = time_delay(xs,xr1,xr2,c)

% function td = time_delay(xs,xr1,xr2,c)
% INPUTS
%     xs - three-dim. vector of source position
%     xr1 - three-dim. vector of receiver 1 position
%     xr2 - three-dim. vector of receiver 2 position
%     c  - speed of sound
%
% Computes the time delay of arrival of a signal from one source to two
% different receiving sensors. (TOA sensor 1 - TOA sensor 2).
%
% Created: 27 June 1991, Ben Rosen, Atlantic Aerospace Electronics Corp.
% Last modified: 1 July 1991, Ben Rosen.
%
t1 = travel_time(xs,xr1,c);
t2 = travel_time(xs,xr2,c);

td = t1-t2;
%keyboard;

```

```
#!/bin/csh -fx
#PROCESS -script
#Executive routine for time-delay estimation analysis.
set SNR = 1
set BW = 2000
set FS = 100000

goto $1

A:
make_noise
exit(0)
B:
make_signal $SNR $BW $FS
#exit(0)

C:
add wgn_sens1 signal1 sn1
add wgn_sens2 signal2 sn2

set D = `echo "$BW/$FS" | bc -l`

peak_correl sn1 sn2 200 $D

exit(0)
```

```

#!/bin/csh -fx
#MAKE_NOISE - script
# This script file generates two 200000-length gaussian white noise sequences.
# It does this by generating a single 400000-length g.w.n sequence and
# splitting it into two equal halves. 200000 samples represents 20 second at
# 10 KHz Sampling Rate. This will represent the sampled gaussian noise at two
# sensors. It uses the MATLAB random number generator since I don't think the
# wgn004 generator used in SYNTH has a long enough period.

set outfile1 = 'wgn_sens1'
set outfile2 = 'wgn_sens2'
set nexp = 200 #Number of Experiments (.1 seconds each)
set nsampc = 1000 #Number of samples per experiment
set nsamps = `echo "$nexp*$nsampc" | bc -l` #Number of sample per sensor
set samp_rate = .0001
set sd = 0
set nsamps2 = `echo "$nsamps *2 " | bc -l`

cat <<EOF >tempo

rand('normal');
rand('seed', $sd);
N = $nsamps2;
R = rand(N,1);
sd = rand('seed')
save sd sd
save R R;

EOF

#invoke MATLAB
matlab < tempo

rm tempo
matdat R rnd

dat_change_dims rnd $nsamps2
rm R.mat #Big file so get rid of it
matdat sd sdl
pick sdl seed -D
rm sd
rndat sdl

set ss = `prdat -n seed`
set sd = $ss[2]

datkey_sampling_period=".0001" rnd
pick rnd $outfile1 0+$nsamps -f
pick rnd $outfile2 $nsamps+$nsamps -f
rndat rnd
#channelize outfiles to do multiple experiments
dat_change_dims -c $nexp $outfile1 $nsampc
dat_change_dims -c $nexp $outfile2 $nsampc
exit(0)

```

```

#!/bin/csh -fx
#MAKE SIGNAL - script
set SNR = $1
set BW = $2
set FS = $3
set infile = wgn
cat <<EOF >ntemp.m

[a,b] = filt_coef($SNR,$BW,$FS);
A = [a b];
save A A;

EOF
matlab <ntemp.m
matdat A A.dat
pick A.dat a.dat 0+1
pick A.dat b.dat 1+1
rmdat A.dat
rm A.mat
rm ntemp.m

#Load filter Coefficients into dat-files
#echo "0.83"> a #5000 Hz BW
#echo "0.964"> a #1000 Hz BW
#asc2dat -f - a a.dat

#FOR 1000 HZ
#echo ".836" > b #About 10 db
#echo ".470" > b #About 5 db
#echo ".297" > b #About 1 db
#FOR 5000 HZ
#echo "1.74" > b #About 10 db
#echo ".98" > b #About 5 db
#echo ".62" > b #About 1 db

#asc2dat -f - b b.dat

#rm -rf a
#rm -rf b

set tauc = 'echo "1.0/$FS" | bc -l' #sampling period of continuous signal
set N = 20000 #Two/tenth's second of data

#This just divides by 2 and gets an integer

echo "scale = 0" > dummy
echo "$N/2" >> dummy
echo "quit" >> dummy
set N2 = 'bc -l dummy'
rm dummy

#invoke MATLAB to generate white noise
#get seed
set ss = 'prdat -n seed'
set sd = $ss[2]
#Synthesize White Noise to Pass to Filter

cat <<EOF >tempo

rand('normal');
rand('seed', $sd);
N = $N;
R = rand(N,1);
sd = rand('seed');
save R R
save sd sd

```

EOF

```
#invoke matlab  
matlab < tempo
```

```
rm tempo  
matdat R $infile  
rm R.mat  
dat_change_dims wgn $N  
matdat sd sd1  
rm sd.mat  
pick sd1 seed -D  
rmdat sd1
```

```
# Filter White Noise to Make Colored Noise which is our signal  
iir $infile a.dat b.dat s1  
cpdat s1 s2
```

```
#Shift by 5005 points (about .05 seconds at 100 KHz)  
set tshift = 5005  
cshift $tshift s2 s3  
#zero out first $tshift points in s3  
insert -V 0+$tshift 0.0 s3  
pick s1 s4 0+$N2  
pick s3 s5 0+$N2
```

```
#Decimate by 10 to get 10KHz Sampled Signal  
pick s4 signal1 :/10  
pick s5 signal2 :/10  
rmdat s1  
rmdat s2  
rmdat s3  
rmdat s4  
rmdat s5
```

```
exit(0)
```

```

#!/bin/csh -fx
#PEAK CORREL - script
set nchan = $3

set s1 = $1
set s2 = $2
set rBW = $4
set D = 5005

fft $s1 S1
conjg S1 Slc
rmdat S1
fft $s2 S2
mul Slc S2 2
rmdat S2
rmdat Slc
ifft 2 z
rmdat 2
real z zr
rmdat z
sortdat -r -n 1 -I corr_pk zr pkval

#Create dat-file to contain interpolated peaks
insert -C $nchan -f 0 . interp_pk

set indx = 0
while ( $indx < $nchan )
    pick zr zrtemp ch {$indx}:{$indx}
    pick corr_pk ctemp ch {$indx}:{$indx}
    set vx = 'prdat -n ctemp'
    set pkindx = $vx[2]
    @ pkstrt = $pkindx - 4
    @ pkstp = $pkindx + 4

    set ninterp = 'echo "($pkstp-$pkstrt)*10 + 1" | bc -l'
    lagrange -o 5 $pkstrt,$pkstp,$ninterp zrtemp zr10
    sortdat -r -n 1 -I int_pk zr10 intpkval
    set vx = 'prdat -n int_pk'
    set int_pkindx = $vx[2]
    set true_pk = 'echo "$pkstrt*10 + $int_pkindx" | bc -l'
    insert -f -V $indx+1 $true_pk interp_pk
    @ indx += 1
end

datnat interp_pk IP
#rmdat interp_pk
rmdat zrtemp zr
rmdat ctemp
rmdat zr10
rmdat intpkval
rmdat int_pk

cat <<EOF >ntemp.m

load IP;
IP(1)={};
[mn,st,per]=delay_stat(IP,$D,$rBW);
mn
st
per

EOF
matlab <ntemp.m
rm ntemp.m
exit(0)

```

%DELAY\_STAT is a routine which computes the mean and standard deviation of a %matrix of values while editing outliers by using a t-sigma cutoff.

```
%  
% [m,s,per]=delay_stat(x,xtrue,rBW)  
% Input  
%   x   Nrow-x-Mcol array  
%   xtrue   True value of x  
% Output  
%   m   lrow-x-Mcol array containing mean of each column  
%   s   lrow-x-Mcol array containing std. dev. of each column  
%   per lrow-x-Mcol array containing percent of each column satisfying  
%       criterion.  
%  
% Created 07 October, 1991 by Ben Rosen  
% Atlantic Aerospace Electronics Corporation  
%
```

```
function [m,s,per]=delay_stat(x,xtrue,rBW);
```

```
t = 1.0/rBW/2;  
dx = abs((x - xtrue));  
inliers = find(dx <= t);  
outliers = find(dx > t);  
xi = x(inliers);  
xo = x(outliers);  
mask = (dx <= t);  
m= mean(xi);  
s=std(xi);  
per=mean(mask)*100;
```